DEPARTAMENT OF SIGNAL PROCESSING AND COMMUNICATIONS

UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TELECOMMUNICATIONS ENGINEERING

FINAL YEAR PROJECT

CLUSTERING TECHNIQUES FOR BASE STATION COORDINATION IN A WIRELESS CELLULAR SYSTEM

AUTHOR: FELIPE LLINARES LÓPEZ
SUPERVISORS: MATILDE PILAR SÁNCHEZ FERNÁNDEZ
EMILIO PARRADO HERNÁNDEZ

17 September 2012

Título:	CLUSTERING TECHNIQUES FOR BASE-STATION COORDINATION IN A WIRELESS CELLULAR SYSTEM
Autor:	FELIPE LLINARES LÓPEZ
Supervisores:	MATILDE PILAR SÁNCHEZ FERNANDEZ Y EMILIO PARRADO HERNÁNDEZ

La defensa del presente Proyecto Fin de Carrera se realizó el día 17 de Septiembre de 2012; siendo calificada por el siguiente tribunal:

Presidente:

Secretario

Vocal

Habiendo obtenido la siguiente calificación:

Calificación:

Presidente **Secretario** **Vocal**

Agradecimientos

En primer lugar, querría agradecer a mis tutores Mati y Emilio por haberme dado la oportunidad de llevar a cabo este proyecto, que no sólo me ha permitido aprender un montón de cosas interesantes sino también dar unos primeros pasos en el mundo de la investigación. Además, también me gustaría felicitarles por ser de los mejores profesores que hemos tenido a lo largo de la carrera, felicitación que hago extensiva al subconjunto de profesores de la Universidad Carlos III de Madrid que deciden dedicar parte del tiempo que podrían destinar a sus labores de investigación a lograr que su actividad docente sea verdaderamente buena y útil para sus alumnos. Formar a las futuras generaciones es algo esencial para la sociedad y, por tanto, me parece que dichas personas constituyen un ejemplo a seguir para el resto del personal docente.

Por otro lado, quiero agradecer a mis padres María Jesús y Felipe todo lo que han hecho por mí durante toda mi vida. Siempre habéis cuidado de mí, quizás demasiado, a pesar de las dificultades que hayan podido surgir. Sois los mejores padres que uno puede desear.

Finalmente, doy las gracias de todo corazón a mis compañeros de clase, por todos las dificultades que hemos superado juntos a lo largo de estos cinco años y, más especialmente, a aquellas personas con quién he tenido la enorme suerte de compartir algunos momentos de mi vida y que tanto me han ayudado, sobretodo cuando menos lo merecía y sin pedir nada a cambio.

A todos, **gracias**.

Resumen

A lo largo de este Proyecto Fin de Carrera, propondremos mejoras para futuros sistemas de comunicaciones móviles mediante un estudio detallado de la coordinación entre estaciones base en sistemas celulares basados en MIMO. Este proyecto se compone de dos partes fundamentales.

Por un lado, nos centraremos en técnicas de procesado de señal para MIMO como filtrado y precodificación lineales en el dominio espacial. Partiendo de los últimos desarrollos en dicho ámbito, se han desarrollado precodificadores de mínimo error cuadrático medio que incluyen restricciones de máxima potencia transmitida por celda. Además, se ha propuesto un concepto novedoso consistente en la introducción de una nueva formulación que, además de minimizar el error cuadrático medio en el interior de cada agrupación de celdas (*cluster*), trata de mantener la interferencia entre *clusters* en niveles suficientemente bajos.

Durante la segunda parte, analizaremos el impacto que la agrupación de celdas en *clusters*, que define qué estaciones base pueden ser coordinadas entre sí, tiene en el rendimiento global del sistema. Se ha estudiado la aplicabilidad de técnicas de agrupamiento dentro del aprendizaje máquina, dando como resultado un conjunto de nuevos algoritmos que han sido desarrollados adaptando algoritmos de agrupamiento de propósito general ya existentes al problema de crear una partición del conjunto de celdas de acuerdo a las condiciones de propagación de señal existentes en el sistema en un determinado instante.

Todas nuestras contribuciones se han verificado mediante la simulación de un sistema de comunicaciones móviles basado en modelos de propagación de señal del 3GPP para LTE. De acuerdo a los resultados obtenidos, las técnicas propuestas a lo largo de este proyecto proporcionan un aumento considerable de la media y la mediana de las tasas por usuario respecto a soluciones ya existentes.

La idea de introducir la reducción de interferencia entre *clusters* en la formulación de los precodificadores MMSE mejora dramáticamente el rendimiento en sistemas celulares MIMO al ser comparados con precodificadores de Wiener tradicionales.

Por otro lado, nuestros algoritmos de agrupamiento dinámico de estaciones base exhiben un notable aumento de las tasas por usuario a la vez que emplean *clusters* de menor tamaño con respecto a soluciones existentes basadas en particiones estáticas del conjunto de celdas en el sistema.

Abstract

In this project, we attempt to provide enhancements for future mobile communications systems by carrying out a throughout study of base-station coordination in cellular MIMO systems. Our work can be divided in two main blocks.

During the first part, we focus our attention on linear MIMO signal processing techniques such as linear spatial precoding and linear spatial filtering. Starting from the state-of-the-art in that area of knowledge, we have developed novel MMSE precoders which include per-cell power constraints and a new formulation which, apart from minimizing the intra-cluster MSE, tries to keep inter-cluster interference at low levels.

In the second part, we focus on the study of the impact the particular mapping of cells to clusters in the cellular system has on the overall performance of the mobile communication radio access network. The applicability of existing clustering algorithms in the field of machine learning has been studied, resulting in a set of novel algorithms that we developed by adapting existing general-purpose clustering solutions for the problem of dynamically partitioning a set of cells according to the instantaneous signal propagation conditions.

All our contributions have been exhaustively tested by simulation of a cellular mobile communication system based on 3GPP signal propagation models for LTE. According to the results obtained, the techniques proposed along this project provide a remarkable increase of both the average and median user rates in the system with respect to previous existing solutions.

The inter-cluster interference-awareness we introduced in the formulation of MMSE precoders dramatically increases the performance in cellular coordinated MIMO when comparing it with traditional Wiener precoders.

On the other hand, our dynamic base-station clustering has been shown to significatively enhance the user rates while using smaller clusters that existing solutions based on static partitions of the base-station deployment.

Contents

List of Figures

List of Tables

List of Algorithms

List of abbreviations

AGC: Automatic Gain Control

BD: Block Diagonalization

BTS: Base Transceiver Station

CDF: Cumulative Density Function

EVOL-MS-BTS-C: EVOLutionary Mean Shift base-station Clustering

GEN-BTS-C: GENetic base-station Clustering

GSM: Global System for Mobile Communications

HAC: Hierarchical Agglomerative Clustering

HDSC: Hierarchical Divisive Spectral Clustering

KKT: Karush-Kuhn-Tucker

LTE: Long Term Evolution

MIMO: Multiple Input Multiple Output

MF: Matched Filter

MS-BTS-C: Mean Shift base-station Clustering

MSE: Mean Square Error

MMSE: Minimum Mean Square Error

PAPC: Per-Antenna Power Constraint

PBPC: Per-BTS Power Constraint

PDFP: robability Density Function

RKHS: Reproducing Kernel Hilbert Space

SC: Spectral Clustering

SNR: Signal-to-Noise Ratio

SNIR: Signal-to-Noise-and-Interference Ratio

SVD: Singular Value Decomposition

SVM: Support Vector Machine

UE: User Equipment

WF: Wiener Filter

ZF: Zero Forcing

3GPP: Third Generation Partnership Project

Notation

We are going to begin by introducing the notation we will use for the remainder of this document when dealing with matrices and vectors defined by blocks. Let us consider an arbitrary matrix $\mathbf{A} \in \mathbb{C}^{Mm \times Nn}$. Such a matrix lends itself to be partitioned in blocks of size $m \times n$ as:

$$\mathbf{A} = \begin{pmatrix} (\mathbf{A})_{1,1} & (\mathbf{A})_{1,2} & \dots & (\mathbf{A})_{1,N} \\ (\mathbf{A})_{2,1} & (\mathbf{A})_{2,2} & \dots & (\mathbf{A})_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{A})_{M,1} & (\mathbf{A})_{M,2} & \dots & (\mathbf{A})_{M,N} \end{pmatrix} \tag{1}$$

Where $(\mathbf{A})_{i,j}$ refers to the block situated in the i-th row and j-th column, understanding now row and column in a block-wise manner. Each of those blocks are themselves $m \times n$ matrices of the form:

$$(\mathbf{A})_{i,j} = \begin{pmatrix} (\mathbf{A})_{i,j}^{1,1} & (\mathbf{A})_{i,j}^{1,2} & \dots & (\mathbf{A})_{i,j}^{1,n} \\ (\mathbf{A})_{i,j}^{2,1} & (\mathbf{A})_{i,j}^{2,2} & \dots & (\mathbf{A})_{i,j}^{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{A})_{i,j}^{m,1} & (\mathbf{A})_{i,j}^{m,2} & \dots & (\mathbf{A})_{i,j}^{m,n} \end{pmatrix} \tag{2}$$

In this case, we use $(\mathbf{A})_{i,j}^{k,l}$ to refer to the individual element in the k-th row and l-th column of the block $(\mathbf{A})_{i,j}$. The reader can check that, under this definition, the element $(\mathbf{A})_{i,j}^{k,l}$ will actually be situated in the $(k + (i-1)m)$-th row and $(l + (j-1)n)$-th column of the complete matrix $\mathbf{A}$. In a similar way, when we have a vector $\mathbf{a} = \begin{bmatrix} \mathbf{a}_1^T \mathbf{a}_2^T \dots \mathbf{a}_N^T \end{bmatrix}^T$ we will refer to particular elements within each block $\mathbf{a}_i$ as $\mathbf{a}_i^j$. To avoid confusion in the future, throughout this project we will index particular elements within a matrix using parenthesis and superscripts, as in $(\mathbf{A})^{i,j}$. Similarly, we will index the k-th element of a vector $\mathbf{a}$ as $\mathbf{a}^k$. The only exceptions to those rules will occur for vectors and matrices whose elements have been assigned specific symbols because they bear a specially relevant meaning for the discussion.

Alternatively, we will also define:

$$(\mathbf{A})_i = \begin{pmatrix} (\mathbf{A})_{i,1} & (\mathbf{A})_{i,2} & \dots & (\mathbf{A})_{i,N} \end{pmatrix} \tag{3}$$

And:

$$(\mathbf{A})^j = \begin{pmatrix} (\mathbf{A})_{1,j} \\ (\mathbf{A})_{2,j} \\ \vdots \\ (\mathbf{A})_{M,j} \end{pmatrix} \tag{4}$$

By doing that, we have a compact way of referring to a set of m rows or n columns of a matrix $\mathbf{A}$. Moreover, if the values of m and/or n were not absolutely clear from the context, we would specify them prior to any further discussion.

The previous notation is to be used in partitions for which all blocks have the same size. When that is not the case, we will denote a partition of an arbitrary matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$ as:

$$\mathbf{A} = \begin{pmatrix} (\mathbf{A})_{m_1 \times n_1} & (\mathbf{A})_{m_1 \times n_2} & \dots & (\mathbf{A})_{m_1 \times n_b} \\ (\mathbf{A})_{m_2 \times n_1} & (\mathbf{A})_{m_2 \times n_2} & \dots & (\mathbf{A})_{m_2 \times n_b} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{A})_{m_a \times n_1} & (\mathbf{A})_{m_a \times n_2} & \dots & (\mathbf{A})_{m_a \times n_b} \end{pmatrix} \tag{5}$$

Where the subscript notation with $\times$ refers to the block size, that is, $(\mathbf{A})_{m_i \times n_j} \in \mathbb{C}^{m_i \times n_j}$. In this way, any two sets $\{m_i\}_{i=1}^{a}$, $\{n_j\}_{j=1}^{b}$ such that $\sum_{i=1}^{a} m_i = m$ and $\sum_{j=1}^{b} n_j = n$ uniquely determine a partition of $\mathbf{A}$. Also, with this notation we can then refer to any block in the partition without ambiguity provided that all blocks have different sizes, which is precisely the type of situation for which this terminology will be used.

On a different matter, we will denote sets by blackboard bold typeface. When defining the elements contained in the set, we will use $\mathbb{A} = \{a_1, a_2, \dots, a_n\}$ or $\mathbb{A} = \{a_i\}_{i=1}^{n}$ without distinction. The cardinality of a set, that is, the number of elements contained by the set, will be denoted as $|\mathbb{A}|$.

Finally, round(x) is defined as the function which rounds x to the nearest integer.

Chapter 1

Introduction

This project focuses on next generation mobile communication systems, such as the 4G and 4.5G standards. In those systems, there is a vast amount of radically different features present at every level of the protocol stack. Within the physical layer, we concentrate on one of the most innovative improvements which will be included in the radio access network in the future: base-station coordination. By allowing several cells to work together, jointly generating the signal to be transmitted and sharing information about the signal which has been received, we introduce a considerable diversity gain. More importantly, cellular coordination is the key to removing or, at least, reducing inter-cell interference in mobile communication systems, as we will discuss next.

1.1 Motivation

The evergrowing bandwidth demand in the radio access network, necessary to support services which are essential nowadays such as real-time video streaming, poses a fascinating challenge for communication engineers: the wireless spectrum is extremely saturated and current modulation schemes are already excellent. At this point, there is no other option than trying to push spectral efficiency towards ridiculously high levels.

As a consequence, Multiple Input Multiple Output (MIMO) communications have stepped up from a secondary role in 3G systems to being a fundamental player in future cellular communications: without MIMO, it would simply be impossible to satisfy the throughput specifications for 4G and 4.5G systems.

Focusing on OFDM based standards such as 4G or 4.5G, since each Orthogonal Frequency Division Multiplex (OFDM) slot can be processed in parallel, we can consider that any given cell will serve one user at each particular central frequency. Multiple access thus occurs as frequency multiplexing by statistical sharing of OFDM carriers in each cell. Precisely because transmissions occurring at different OFDM carriers are orthogonal, the system can be thought as a set of M cells serving a set of M users, one per cell.

Initially, signal processing was applied individually for each BTS-user pair. In other words,

the complete system behaves as a set of many parallel point-to-point MIMO subsystems. However, even though we can consider that interference between users in the same cell is negligible thanks to OFDM, current throughput requirements force the usage of universal frequency reuse and, as a consequence, interference between cells does exist. In other words, those M point-to-point MIMO systems interference with each other, noticeably increasing the noise level their experience and, in the end, limiting the improvement MIMO can offer.

Therefore, even the introduction of MIMO is no longer enough. In a scenario where everything needs to be optimized and with the average cell radii becoming smaller in urban environments as the time progresses, interference between cells is becoming a major problem which needs to be addressed.

This motivates the concept of Base Transceiver Station (BTS) coordination for joint MIMO signal processing. The principle is theoretically simple: several cells are joined into a single cluster and signal processing is then performed in a coordinated manner between all base-stations in the cluster. In this way, MIMO techniques are no longer applied in a per-cell basis but, instead, each cluster is considered as a macro-cell with many distributed transmit and receive antennas. This allows to significatively reduce, or even completely eliminate, interference between cells belonging to the same cluster.

Ideally, we would like to coordinate all BTSs in an area as big as possible in order to provide fully interference-free communications to the terminals. However, such a thing is obviously unfeasible, given the amount of channels to be estimated and base-stations to be coordinated. A trade-off between feasibility and performance appears where BTS coordination is applied but within clusters which contain a relatively small amount of base-stations, thus keeping implementation costs at reasonable levels.

During the last years, MIMO coordinated transmission has been a subject which was embraced with enthusiasm by researchers worldwide. A wide variety of MIMO signal processing techniques specially engineered for this application have been proposed in the literature. However, up to the author's knowledge, a fundamental design parameter has been surprisingly ignored in a systematic way: how the BTSs are grouped to form the clusters. It seems that most authors assume that clusters are obtained via geographical considerations, in a way similar to how clusters were created in Global System for Mobile Communications (GSM) systems to deal with the problem of frequency allocation and frequency reuse.

Motivated by such lack of knowledge regarding the influence of the cluster shapes on the overall system performance, our fundamental ambition was to open new research lines by, first of all, assessing whether it was possible to increase the throughput of a mobile communication system with MIMO base-station coordinated transmission by changing the way clusters are created. Moreover, an algorithm for dynamical allocation of base-stations to clusters aiming to maximize the performance was to be developed in case the answer to the previous question was affirmative.

An extensive review of MIMO signal processing techniques lead us to realize about the

absence of MIMO MMSE linear filtering techniques that were applicable in distributed multi-user scenarios such as any cellular environment. This led to the study and proposition of several novel filtering techniques which try to fill that gap.

On the other hand, finding ways of creating BTS clusters turned out to be much harder than expected. What was supposed to be a "one-algorithm project" ended up as a thorough study, under the field of machine learning, of state-of-the-art clustering techniques and, later, genetic algorithms. Based on those existing algorithms, intended mainly for data-mining applications, we have created a new set of clustering algorithms tailored specifically for tackling the problem of grouping base-stations for coordinated MIMO transmission.

During the rest of this chapter, we will describe the different objectives of this project, the methodology we followed to achieve those goals and the main contributions which resulted from our work. Also, we will briefly describe the organization of the remainder of this document.

1.2 Objectives and contributions

The ultimate goal of this project is to study the effect of changing the way in which cells are grouped to coordinate and to develop algorithms which find optimal mappings of cells to clusters. Nevertheless, given the broad scope of this project, it should not be a surprise that its objectives also encompass a much wider variety of points. In general, we can roughly classify this project's goals in three main groups.

On the one hand, given the multidisciplinary nature of problem at our hands, we had a great deal of work to be done in order to research the state-of-the-art of several seemingly unrelated topics. MIMO communications and their applications to cellular systems with base-station coordination was the first point to be covered. However, it was equally necessary to obtain a solid background in clustering algorithms for data mining. Also, even if it was not expected at the beginning, genetic algorithms became in the end a fundamental part of our work too.

All the previous objectives were directed towards achieving a solid background in the different areas of knowledge which we intend to use for satisfying our ultimate goal. Once such background has been obtained, the following objectives consist of employing the acquired knowledge to develop our contributions from a theoretical point of view. In this sense, we have worked mainly with two different aims: the design of novel MMSE precoders for cellular coordinated MIMO and the creation of base-station clustering algorithms.

Finally, the last step of this project is the implementation in MATLAB of all the software we need for testing our derivations.

First of all, a library which allows us to simulate a mobile communication system with MIMO transceivers and arbitrary base-station coordination has been developed. As a starting point, we had available a simplified library used for previous projects in the Department of Signal Processing and Communications. However, many extra contributions were required to adapt the existing software to our needs. On the one hand, it was necessary to implement from scratch

a complete library to obtain MIMO precoders and receive filters, both those which are popular in the specialized literature and the ones we have created ourselves during this project. On the other hand, we had to extend the existing library to cover different types of simulation scenarios, signal propagation models and, more importantly, to include base-station coordination.

On a different matter, we also needed at an early stage of the project to implement different general-purpose clustering algorithms which are not available as MATLAB toolboxes. Those are mainly spectral clustering, the mean-shift algorithm and, of course, any of the evolutionary clustering algorithms we have considered. Even though we do not use those general-purpose algorithms in the final stage of the project, they were needed as an intermediate step in order to study and comprehend their behavior, which allowed us to particularize them better for our context in cellular coordinated MIMO.

Last but not least, we needed to implement all the base-station clustering algorithms we created by modifying the general-purpose algorithms we had studied before.

Literature review

- Exhaustive review of state-of-the-art MIMO communications with emphasis in linear precoding and filtering techniques.
- Exhaustive review of state-of-the-art clustering algorithms in the field of machine learning.
- Exhaustive review of the state-of-the-art in the application of genetic algorithms for clustering.

Theoretical developments

- Study of new ideas and concepts to design better linear precoders and filters for cellular coordinated MIMO.
- Attempt to introduce per-cell power constraints in the formulation of MMSE MIMO precoders.
- Study the particularities of the problem of grouping base-stations for coordinated MIMO transmission when treating it as a machine learning clustering problem.
- Develop new specialized clustering algorithms to group base-stations for coordinated MIMO transmission in mobile communications.

Software implementations

- Implementation of a library in MATLAB which allows to obtain MIMO linear precoders and filters, both those which are well-known by the literature and those which have developed by us.
- Implementation of a library in MATLAB which allows simulating a wireless cellular system based on MIMO with arbitrary coordination between the distinct cells in the deployment.

- Implementation in MATLAB of the general-purpose clustering algorithms not already included in regular toolboxes: spectral clustering, mean-shift and genetic clustering algorithms.
- Implementation in MATLAB of all the base-station clustering algorithms developed in this project.

The fundamental contribution of this project has been opening a new research line. We wanted to prove that not paying enough attention to the way in which we define with which cells each base-station coordinates is a bad decision. By showing that it is possible to greatly enhance the performance of the system only by carefully designing the clusters which define the coordination within the mobile communication system, we expect that further efforts will be developed in this new area of research.

In this same direction, we have also contributed with a systematic exploration of clustering methods to obtain partitions for such purpose. Some of them, like those based on genetic algorithms, are unfeasible in practice yet they are invaluable from a research point of view. As they are actually able to find the optimal partition for a given scenario, even if they do it slowly, we can use them to further study the problem and design other faster, suboptimal base-station clustering algorithms. We have also developed other algorithms which are feasible, but an efficient tuning of their design parameters remains an open question.

On a slightly different area, we have also produced valuable contributions in the design of MIMO linear precoders. We came up with a new concept, which we denoted interference-awareness. The idea is to design precoders which consider not only the signal propagation within their cluster but which also try to keep the interference levels they generate outside the cluster as low as possible. Empirical simulations have shown that the introduction of this idea leads to an enormous performance increase for cellular coordinated MIMO.

More importantly, we have tackled the seemingly ignored problem of including per-cell power constraints into the formulation of MMSE precoders. Without this type of constraints, it would not be possible to use those schemes in a multi-user distributed MIMO scenario such as any mobile communication network. Even though we were not able to obtain a closed form solution, our derivations contain several really interesting facts about the nature of the problem. That could represent a major contribution as the interference-aware MMSE precoder has been shown to hold the potential to even surpass block diagonalization in performance.

1.3 Document structure

The first chapters try to present the framework we will need later in this project. Once all the foundations have been presented, we discuss in great detail our contributions from a theoretical point of view. The document ends by presenting and discussing empirical results which shed light on the behavior and characteristics of the algorithms and signal processing techniques

which resulted from our work. The main contents of each chapter are described next in a deeper detail.

Chapters 2 and 3 are devoted to developing from scratch all the required background in MIMO communications. During chapter 2, we define what we understand by a MIMO communication system; explain the different particularizations of the general model, focusing on their relation with mobile communications; and discuss the most important performance measures associated to MIMO systems. On the other hand, in chapter 3 we start with a self-contained review of the state-of-the-art in MIMO linear precoding and filtering to end up presenting our own contributions in that area of knowledge, which are mainly focused on novel MMSE linear precoders for cellular coordinated MIMO.

Chapter 4 is an introduction to clustering from a machine learning perspective. First, the concept of clustering itself is discussed. After that, traditional algorithms such as K-means or hierarchical clustering are explained. At that point, we start dealing with more advanced algorithms such as spectral-clustering and the mean-shift procedure. Both are developed from scratch without assuming any previous knowledge. Finally, we also provide an introduction to genetic algorithms and present the main guidelines for applying evolutionary computation to clustering problems. Appendices A and B have been included with specific information about some of the clustering algorithms described in chapter 4. The content of those appendices provides useful a quite useful insight yet it is not fundamental for understanding this project.

Chapter 5 begins analyzing which are the exact characteristics of our problem with regards to clustering. By using the intuitive arguments we developed, we continue by explaining each of the base-station clustering algorithms we have created along this project, which are inspired on those shown in chapter 4. Their pseudo-code and parametrization is also discussed.

The simulation results for our work are divided between chapter 6 and appendix C. In chapter 6, we explain the main simulation scenario, include the particular tables and graphs holding the figures and make a detailed discussion of the results. On the other hand, appendix C is devoted to discussing one of the particular ideas we introduced for MIMO MMSE precoders which we considered to be really interesting, yet it could not be included in the main simulations for reasons which will become clear later on this project.

The document ends with chapter 7, where we summarize what has been done along this project, make a critical review of what we have accomplished from an objective perspective and hint at the main issues and topics we have left open for further research.

Chapter 2

State-of-the-art MIMO systems

Wireless communication systems are required to provide higher data rates as the time goes by. In the past, the throughput of the system was usually risen either by using more bandwidth or employing better modulations. However, because those two ideas are starting to reach their inherent limitations at our current level of technological development, we need new methods to improve the performance of the system.

One of the most popular derivations to deal with that issue are the so called Multiple Input Multiple Output (MIMO) systems. Those employ multiple antennas in the transmitter and/or receiver and have been shown to be able to obtain significant enhancements of the data rate of the system without increasing the total transmitted power.

During the remainder of this chapter, we will present a detailed mathematical model of MIMO communication systems and study some of their peculiarities.

2.1 Generic system model

We will consider a MIMO system with N users equipped with r receive antennas and M BTS with t transmit antennas each. Therefore, we have a distributed MIMO system with Mt transmit antennas and Nr receive antennas. For most tx-rx designs we need that $Mt > Nr$ since otherwise we would lose entire data flows in the transmission.

The reader can note that the previous paragraph describes a downlink channel where the BTSs act as transmitters and the users act as receivers. However, practically all the derivations done for the downlink channel for this project will still hold for the uplink channel just by swapping the words BTS and users in the text. Because of this, we will keep on using a notation derived for the downlink channel and, unless otherwise noted, the reader can consider that the results apply without change to the uplink also.

We assume that the signal will propagate through an environment with lots of reflectors and scatterers, as it happens for instance on any urban scenario. We also consider that we have no ISI in the system. As we will point out later, this supposition is more general than it seems since it holds true for any narrow band system (those are generally subject to slow and flat fading)

and, therefore, applies for any OFDM system on a per-carrier basis. As a consequence of our initial hypothesis, we will model the channel with attenuation due to path-loss and shadowing plus Rayleigh fading. Because of that, the channel will be represented by a matrix $\mathbf{H} \in \mathbb{C}^{Nr \times Mt}$ so that $(\mathbf{H})_{i,j}^{k,l}$ represents the total attenuation and the phase shift between the signal sent by the l-th transmit antenna in the j-th base-station and the signal received at the k-th receive antenna of the i-th user.

In the most general case, signal processing is carried out both by the transmitters and the receivers. In order to satisfy a trade-off between system complexity and performance, we will only consider linear filtering along this project. The system will be designed to transmit K data streams in parallel, which are grouped in the vector $\mathbf{u} \in \mathbb{C}^K$. Along this project, the data symbol vector $\mathbf{u}$ is considered a random vector with arbitrary autocorrelation matrix $\mathbf{R_u}$, even though most formulations in the literature consider $\mathbf{u}$ to be a white (both spatially and in the time domain) Gaussian vector with unit power, that is, $\mathbf{R_u} = \mathbf{I}$. That vector is linearly precoded by the transmit filter $\mathbf{W_{tx}} \in \mathbb{C}^{Mt \times K}$ so that the set of symbols sent by the base-stations is contained in the vector $\mathbf{x} = \mathbf{W_{tx}u} \in \mathbb{C}^{Mt}$.

The received signal is a vector $\mathbf{y} \in \mathbb{C}^{Nr}$ which is the output of the channel $\mathbf{H}$ when it is excited by the transmitted signal vector $\mathbf{x}$, plus an additive noise term modeled by the vector $\mathbf{n} \in \mathbb{C}^{Nr}$. Even though the most usual assumption is that the noise power is identical in all the Nr receive antennas that noise samples corresponding to different antennas are uncorrelated, we will assume for now a more general scenario where $\mathbf{n}$ is a white Gaussian stochastic process with arbitrary autocorrelation matrix $\mathbf{R_n} \in \mathbb{C}^{Nr \times Nr}$. This latter formulation allows an arbitrary correlation between the noise samples of the different antennas in the system. In the end, the channel output can be written as:

$$\mathbf{y} = \mathbf{Hx} + \mathbf{n} = \mathbf{HW_{tx}u} + \mathbf{n} \tag{2.1}$$

Finally, the received signal is processed by the receiver using a linear filter $\mathbf{W_{rx}} \in \mathbb{C}^{K \times Nr}$ so that, at the filter output, we have an estimate of the original data vector $\mathbf{u}$. Mathematically, $\hat{\mathbf{u}} = \mathbf{W_{rx}y}$:

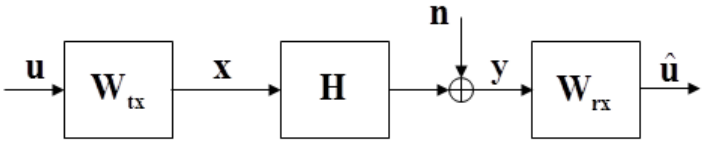


Figure 2.1: General MIMO system model

The end-to-end system can be fully characterized by the equation:

$$\hat{\mathbf{u}} = \mathbf{W_{rx}HW_{tx}u} + \mathbf{W_{rx}n} \tag{2.2}$$

We must note that the maximum number of parallel data streams we can transmit equals

the rank of the channel matrix $\mathbf{H}$. Because of that, we have that $K \leq \min(Mt, Nr)$ which, by the initial hypothesis that $Mt \geq Nr$ implies $K \leq Nr$. However, the channel matrix having a rank strictly smaller than Nr is very rare. Because of that, we will assume from now on that $K = Nr$ for notational simplicity unless otherwise noted.

We can think of the previous system model as being at "antenna level". However, we must note that the vectors $\mathbf{u}$, $\mathbf{x}$ and $\mathbf{y}$ have, in general, the information of N different users. Therefore, it's also interesting to develop a model that tries to show the different subchannels at "user level". We will consider the following partition of the vectors and matrices representing the MIMO system model.

The symbol vector $\mathbf{u}$ is arranged as:

$$\mathbf{u} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_N \end{pmatrix} \tag{2.3}$$

With $\mathbf{u}_i \in \mathbb{C}^r \ \forall\, i = 1, \ldots, N$ being the set of symbols to be transmitted in the r data streams of the i-th user.

The precoder matrix $\mathbf{W_{tx}}$ is partitioned in the following way:

$$\mathbf{W_{tx}} = \begin{pmatrix} (\mathbf{W_{tx}})_{1,1} & (\mathbf{W_{tx}})_{1,2} & \cdots & (\mathbf{W_{tx}})_{1,N} \\ (\mathbf{W_{tx}})_{2,1} & (\mathbf{W_{tx}})_{2,2} & \cdots & (\mathbf{W_{tx}})_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{W_{tx}})_{M,1} & (\mathbf{W_{tx}})_{M,2} & \cdots & (\mathbf{W_{tx}})_{M,N} \end{pmatrix} \tag{2.4}$$

In this case, $(\mathbf{W_{tx}})_j \in \mathbb{C}^{t \times Nr}$ would represent the portion of the precoder which is responsible for obtaining the t symbols to be transmitted from the j-th base-station. Similarly, $(\mathbf{W_{tx}})^i \in \mathbb{C}^{Mt \times r}$ is the portion of the precoder which carries information on the r data streams of the i-th user. Finally, $(\mathbf{W_{tx}})_{j,i} \in \mathbb{C}^{t \times r} \ \forall\, j = 1, \ldots, M \quad i = 1, \ldots, N$ is the block of the precoding matrix which defines the contribution of the r data streams of the i-th user to the t symbols to be sent from the j-th base-station.

The transmitted signal vector $\mathbf{x}$ is split in the following pieces:

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_M \end{pmatrix} \tag{2.5}$$

Where $\mathbf{x}_j \in \mathbb{C}^t \ \forall\, j = 1, \ldots, M$ represents the set of symbols sent by the j-th base-station.

The channel matrix $\mathbf{H}$ is partitioned in a similar way as $\mathbf{W_{tx}}$ but taking into account that the dimensions are inverted:

$$\mathbf{H} = \begin{pmatrix} (\mathbf{H})_{1,1} & (\mathbf{H})_{1,2} & \dots & (\mathbf{H})_{1,M} \\ (\mathbf{H})_{2,1} & (\mathbf{H})_{2,2} & \dots & (\mathbf{H})_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{H})_{N,1} & (\mathbf{H})_{N,2} & \dots & (\mathbf{H})_{N,M} \end{pmatrix} \tag{2.6}$$

The interpretation of the previous partition is similar to that of $\mathbf{W_{tx}}$. $(\mathbf{H})_i \in \mathbb{C}^{r \times Mt}$ stands for the complete channel seen by the i-th user from all the base-stations in the system. On the contrary, $(\mathbf{H})^j \in \mathbb{C}^{Nr \times t}$ would be the channel seen by the j-th base-station towards all the users in the network. Finally, $(\mathbf{H})_{i,j} \in \mathbb{C}^{r \times t} \; \forall \, i = 1, \dots, N \quad j = 1, \dots, M$ is the channel seen between the j-th base-station and the i-th user.

The noise vector $\mathbf{n}$ is partitioned as:

$$\mathbf{n} = \begin{pmatrix} \mathbf{n}_1 \\ \mathbf{n}_2 \\ \vdots \\ \mathbf{n}_N \end{pmatrix} \tag{2.7}$$

With $\mathbf{n}_i \in \mathbb{C}^r \; \forall \, i = 1, \dots, N$ being the noise at the r receive antennas of the i-th user.

The received signal vector $\mathbf{y}$ is arranged in the same way as the noise vector, given that their dimensions are identical:

$$\mathbf{y} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_N \end{pmatrix} \tag{2.8}$$

Where $\mathbf{y}_i \in \mathbb{C}^r \; \forall \, i = 1, \dots, N$ represents the set of symbols received by the i-th user.

The receive filter $\mathbf{W_{rx}}$ is a square matrix, as we assumed that the number of data streams in the system would equal the number of receive antennas. Therefore the partition of $\mathbf{W_{rx}}$ becomes:

$$\mathbf{W_{rx}} = \begin{pmatrix} (\mathbf{W_{rx}})_{1,1} & (\mathbf{W_{rx}})_{1,2} & \dots & (\mathbf{W_{rx}})_{1,N} \\ (\mathbf{W_{rx}})_{2,1} & (\mathbf{W_{rx}})_{2,2} & \dots & (\mathbf{W_{rx}})_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{W_{rx}})_{N,1} & (\mathbf{W_{rx}})_{N,2} & \dots & (\mathbf{W_{rx}})_{N,N} \end{pmatrix} \tag{2.9}$$

The portion $(\mathbf{W_{rx}})_i \in \mathbb{C}^{r \times Nr}$ of the receive filter is used to compute the estimate of the set of symbols for the i-th user. On the other hand, $(\mathbf{W_{rx}})^i \in \mathbb{C}^{Nr \times r}$ represents the contribution of the signal $\mathbf{y}_i$ received by the r antennas of the i-th user on the complete estimate $\hat{\mathbf{u}}$ of the set of all symbols sent by the network. In this way, $(\mathbf{W_{rx}})_{i,j} \in \mathbb{C}^{r \times r} \; \forall \, i = 1, \dots, N \quad j = 1, \dots, N$ is

the contribution of the signal received by the j-th user for obtaining the estimate of the symbols directed towards the i-th user.

Finally, the most natural partition of the estimated symbol vector $\hat{\mathbf{u}}$ is:

$$\hat{\mathbf{u}} = \begin{pmatrix} \hat{\mathbf{u}}_1 \\ \hat{\mathbf{u}}_2 \\ \vdots \\ \hat{\mathbf{u}}_N \end{pmatrix} \tag{2.10}$$

With $\hat{\mathbf{u}}_i \in \mathbb{C}^r \ \forall i = 1, \ldots, N$ containing the estimate of the r symbols in the data-streams for the i-th user.

As we said, the vector $\mathbf{u}_i$ contains the r data symbols corresponding to the i-th user. We will consider that each of the $\mathbf{u}_i$ has an arbitrary autocorrelation matrix $\mathbf{R}_{\mathbf{u}_i} \in \mathbb{C}^{r \times r}$ but we will also suppose that $\mathbf{u}_i$ and $\mathbf{u}_j$ will be uncorrelated for all $i \neq j$. In this way, $\mathbf{R}_{\mathbf{u}} \in \mathbb{C}^{Nr \times Nr}$ becomes a block diagonal matrix:

$$\mathbf{R}_{\mathbf{u}} = \begin{pmatrix} \mathbf{R}_{\mathbf{u}_1} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{\mathbf{u}_2} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{R}_{\mathbf{u}_N} \end{pmatrix} \tag{2.11}$$

Each of those data vectors is precoded by the per-user beamforming matrix $(\mathbf{W_{tx}})^i$ so that the total transmit symbol vector $\mathbf{x}$ can be found only as the addition of the contributions of all the beamformed $\mathbf{u}_i$:

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_M \end{pmatrix} = \sum_{i=1}^{N} (\mathbf{W_{tx}})^i \mathbf{u}_i \tag{2.12}$$

Looking at (2.12) we can realize that, under this general setting, to compute the signal sent by the antennas in the j-th BTS, $\mathbf{x}_j$, we actually need to know the data symbols and per-user beamforming matrices of all the users. As we will discuss in greater detail in the next subsection, this is a tricky issue for practical implementations which will have to be dealt with.

The signal received at the i-th receiver, $\mathbf{y}_i$ can be found as:

$$\mathbf{y}_i = (\mathbf{H})_i \mathbf{x} + \mathbf{n}_i \tag{2.13}$$

As we see, as far as the i-th user goes, only the portion $(\mathbf{H})_i$ of the channel matrix affects the received signal. However, the signal sent by all the transmit antennas contributes to the received signal. The vector $\mathbf{n}_i$ is the noise present at each of the r receive antennas that the i-th user has and is described by its autocorrelation matrix, which is nothing but the i-th, $r \times r$ diagonal block matrix of the autocorrelation matrix of $\mathbf{n}$, $\mathbf{R}_{\mathbf{n}}$.

Finally, the i-th receiver ideally constructs the estimate of the data symbol of the i-th user, $\hat{\mathbf{u}}_i$ as:

$$\hat{\mathbf{u}}_i = (\mathbf{W_{rx}})_i \mathbf{y} \tag{2.14}$$

Again, (2.14) shows that, in order for the i-th receiver to compute the estimate of the data symbol $\hat{\mathbf{u}}_i$, it needs to have access to the signal received by all the antennas of the system, even those corresponding to other users. Just as it happened for constructing the signal to be fed to the transmit antennas, this is impractical in many real-life cases and we will discuss how to face that too in the following subsection.

As we can see, these "user level" derivations indeed show that there is, a priori, no way of completely uncoupling the different users so that the signal received by the i-th user depends only on the i-th data symbol vector $\mathbf{u}_i$.

If we do not apply any kind of signal processing, that is, we don't use any precoder $\mathbf{W_{tx}}$ nor receive filter $\mathbf{W_{rx}}$, all the data-streams are coupled due to the channel matrix $\mathbf{H}$.

Alternatively, we will study in subsequent sections some filter designs which actually attempt (and succeed) in generating a system so that the chain $\mathbf{W_{rx}HW_{tx}}$ behaves as a set of N parallel, non-interacting subchannels. However, equations (2.12) and (2.14) show that computing the signal vector $\mathbf{x}$ to be transmitted by the set of base-stations and obtaining the data symbol estimate $\hat{\mathbf{u}}$ from the signal received by the user antennas requires some sort of coordination between base-stations and/or between users. This is clear from the fact that to compute $\mathbf{x}_j$ and $\hat{\mathbf{u}}_i$ we need to know the complete vectors $\mathbf{u}$ and $\mathbf{y}$.

Therefore, either we have coupling at the signal propagation level or at the signal processing level. The first will significatively decrease the system's performance, whereas the latter increases the complexity and thus the implementation cost. Finding a proper trade-off between both is a fundamental part of the work we carry out during this project.

2.1.1 Canonical MIMO scenarios

The model that has been exposed up to now is a completely general representation of a MIMO scenario with linear filtering. This includes scenarios as different as a communication between one transmitter with multiple antennas to one receiver with multiples antennas or a communication between multiple transmitters with multiple antennas to multiple receivers with multiple antennas.

However, at this point, it is interesting to show how we can particularize it to faithfully represent some of the most usual real-life scenarios a communications engineer may face.

The first and simplest scenario we can think of is what we will call the single-user case. In this situation, we consider that all there is just one single transmitter and one single receiver. According to our model, this is fully equivalent to having $N = 1$ and $M = 1$. Not only this is the simplest MIMO system we can study, but the current state-of-the-art can be considered to

be such that the design of the precoding matrix and the receive filter under this conditions has been fully solved.

Taking a step towards a greater complexity, we can consider either a multi user scenario where a single transmitter, $M = 1$, serves several users, $N > 1$ or a multi user scenario where several users (transmitters), $M > 1$, transmit to the same receiver, $N = 1$. Both scenarios are representative of a situation where a single BTS serves several users: the first case would correspond to the downlink channels whereas the latter models the uplink channel. We will denote this situation as multi-user case.

Finally, the most complex and general situation is to have several transmitters serving several users simultaneously, so that both $M > 1$ and $N > 1$. This will be the case we will end up dealing with along this project. We call this a distributed multi-user case.

The main differences between scenarios revolve around two issues: the number of degrees of freedom we have for designing the filters $\mathbf{W_{tx}}$, $\mathbf{W_{rx}}$ and the transmit power constraints we have to satisfy.

In order to fully understand those points, let's go back to the expression of the set of symbols sent by the j-th base-station:

$$\mathbf{x}_j = \sum_{k=1}^{N} (\mathbf{W_{tx}})_{j,k}\mathbf{u}_k \tag{2.15}$$

This particularly means that, in order to compute the set of symbols to the transmitted by the j-th base-station, we need to know all the Nr data symbols contained in $\mathbf{u}$. If we have only one transmitter, this obviously poses no problem at all since the transmitter software will have access to all the data. However, if we consider a multi-BTS scenario, depending on the number of BTS it may become unfeasible to coordinate them all. Because of that, in scenarios with several BTS, we may impose constraints to the coordination between the distinct transmitters. As an example, the most restrictive coordination constraint would be that, indeed, no coordination exists between the BTSs. In this case, looking at equation (2.15), this actually means that $(\mathbf{W_{tx}})_{j,i} = 0 \quad \forall i \neq j$. In other words, we would be imposing that $\mathbf{W_{tx}}$ is a block-diagonal matrix, $\mathbf{W_{tx}} = \text{diag}\left((\mathbf{W_{tx}})_{1,1}, \ldots, (\mathbf{W_{tx}})_{M,M}\right)$ where each of the $(\mathbf{W_{tx}})_{i,i} \in \mathbb{C}^{t\times r}$. A softer coordination restriction, which is the one towards which this project will be headed, is to allow a set of L BTSs to coordinate, forming a cluster. In this case, we will impose that $(\mathbf{W_{tx}})_{j,i} = \mathbf{0}$ whenever base-stations i and j belong to different clusters. In the particular case where the BTSs are numbered so that all stations belonging to the same cluster have consecutive indexes, again we will have that $\mathbf{W_{tx}}$ is a block-diagonal matrix. However, it will now be constructed as $\mathbf{W_{tx}} = \text{diag}\left(\left(\mathbf{W_{tx}^{(c)}}\right)_{1,1}, \ldots, \left(\mathbf{W_{tx}^{(c)}}\right)_{S,S}\right)$ where each of the $\left(\mathbf{W_{tx}^{(c)}}\right)_{i,i} \in \mathbb{C}^{Lt\times Lr}$ represent the coordinated precoding matrix for the i-th cluster and S the total number of clusters in the system. As a final note, we point out that by no means is it necessary that L is constant for all clusters. In other words, we may design a system where each cluster contains a different number of BTSs, being the only difference that each of the $\left(\mathbf{W_{tx}^{(c)}}\right)_{i,i}$ will have a different size. In the

same way, arranging indexes in consecutive cluster order is simply a visual artifact to achieve a block diagonal $\mathbf{W_{tx}}$. In a more general case where that does not happen, $\mathbf{W_{tx}}$ will no longer be block diagonal but it will still be a sparse matrix, which is associated to the low implementation cost of the base-station coordination process we are looking for.

Similarly, we can apply a similar reasoning at the receiver side. Let's us write the data symbol estimate at the i-th user as:

$$\hat{\mathbf{u}}_i = \sum_{k=1}^{N} (\mathbf{W_{rx}})_{i,k} \mathbf{y}_k \tag{2.16}$$

Again, we see that the previous equation means that in order to compute the i-th data symbol estimate, we need access to the signal received by all the receive antennas. Actually, the previous reasoning applies to the receive side as well. If we have only a single receiver, this does not suppose any problem. On the other hand, when having multiple receivers like, for instance, a set of mobile phones, it becomes unfeasible to coordinate them and we need to restrict $\mathbf{W_{rx}}$ to block-diagonal shapes as exposed above.

Usually, we can allow a limited coordination between the BTSs, but coordination between users is impractical. Because of that, a duality between the downlink and the uplink arises.

MIMO scenario	Downlink	Uplink
Single-user	Full	Full
Multi-user	Full	Block-diagonal
Distributed multi-user	Block-diagonal	Block-diagonal

Table 2.1: Degrees of freedom for the design of $\mathbf{W_{tx}}$ per MIMO canonical scenario

MIMO scenario	Downlink	Uplink
Single-user	Full	Full
Multi-user	Block-diagonal	Full
Distributed multi-user	Block-diagonal	Block-diagonal

Table 2.2: Degrees of freedom for the design of $\mathbf{W_{rx}}$ per MIMO canonical scenario

To sum up, as far as the design of $\mathbf{W_{tx}}$ and $\mathbf{W_{rx}}$ is concerned, the particular scenario the engineer faces imposes a restriction on which elements of those matrices are allowed to be non-zero, effectively limiting the number of degrees of freedom in the design.

On the other hand, when studying the transmit power constraints, what changes between one scenario or another is the granularity of the constraint. The most usual case is that each transmitter has a maximum power available for the transmission. This implies that, for the single user-scenario and the 1 BTS - N users scenario, we need to consider a Total-Power-Constraint (TPC). On the other hand, for the M BTS - N users scenario, we need to use a finer

detail, employing a Per-BTS-Power-Constraint (PBPC). We will quantify both criteria in the next subsection.

MIMO scenario	Downlink	Uplink
Single-user	TPC	TPC
Multi-user	TPC	PBPC
Distributed multi-user	PBPC	PBPC

Table 2.3: Type of power constraint per MIMO canonical scenario

2.1.2 Transmitted power analysis

The transmitted power is a parameter which is essential to know since practically all systems will have one kind of power constraint or another which we will have to satisfy. Going back to the previous formulation, we said that the set of symbols transmitted by the j-th base-station was $\mathbf{x}_j$ with $\mathbf{x} = \mathbf{W_{tx}u}$. Since $\mathbf{u}$ is a stochastic process modeled by its autocorrelation matrix $\mathbf{R_u}$, the average power transmitted by the l-th transmit antenna of the j-th base-station is given by the l-th diagonal entry of the autocorrelation matrix of the random vector $\mathbf{x}_j$, $\mathbf{R_{x_j}}$ where $\mathbf{R_x}$ can be found as:

$$\mathbf{R_x} = \begin{pmatrix} \mathbf{R}_{\mathbf{x}_1} & \mathbf{R}_{\mathbf{x}_1,\mathbf{x}_2} & \dots & \mathbf{R}_{\mathbf{x}_1,\mathbf{x}_M} \\ \mathbf{R}_{\mathbf{x}_2,\mathbf{x}_1} & \mathbf{R}_{\mathbf{x}_2} & \dots & \mathbf{R}_{\mathbf{x}_2,\mathbf{x}_M} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{\mathbf{x}_M,\mathbf{x}_1} & \mathbf{R}_{\mathbf{x}_M,\mathbf{x}_2} & \dots & \mathbf{R}_{\mathbf{x}_M} \end{pmatrix} = \mathbf{W_{tx}R_uW_{tx}^H} \tag{2.17}$$

Where the terms $\mathbf{R}_{\mathbf{x}_i,\mathbf{x}_j}$ involving the cross-correlation between $\mathbf{x}_i$ and $\mathbf{x}_j$ are not relevant for evaluating the power consumption.

In a similar way, the total power consumed by the j-th BTS, with j ranging from 1 to M, can be found as:

$$P_{\text{tx,j}} = \text{Tr}\left(\mathbf{R}_{\mathbf{x}_j}\right) \tag{2.18}$$

Finally, the total power transmitted by the system can be found as:

$$P_{\text{tx}} = \sum_{j=1}^{M} P_{\text{tx,j}} = \text{Tr}(\mathbf{R_x}) \tag{2.19}$$

According to the distinct canonical scenarios exposed in the previous subsection, we can express a TPC as:

$$P_{\text{tx}} \leq P_{\max} \quad \Leftrightarrow \quad \text{Tr}(\mathbf{R_x}) \leq P_{\max} \tag{2.20}$$

And a PBPC as:

$$P_{\text{tx,j}} \leq P_{\max,\text{j}} \quad \Leftrightarrow \quad \text{Tr}\left(\mathbf{R}_{\mathbf{x}_j}\right) \leq P_{\text{tx,j}} \quad \forall j = 1, \ldots, M \tag{2.21}$$

Finally, we can consider yet another power constraint, more restrictive even that the PBPC: a Per-Antenna-Power-Constraint (PAPC). However, a PAPC is mathematically equivalent to a PBPC with only one transmit antenna per BTS, $t = 1$. Being more concrete, if the reader intends to design a system using a PAPC and $t \geq 1$, he/she can simply consider each BTS to be a cluster of t fully-coordinated BTSs with only one transmit antenna, and apply all the derivations shown here for the case with N users served by M BTSs coordinated in clusters and making $t = 1$ for each one of those fictional BTSs.

In the particular case that the data symbols are spatially uncorrelated, $\mathbf{R_u}$ will be a diagonal matrix, that is, $\mathbf{R_u} = \text{diag}(\mathbf{p_u})$. The vector $\mathbf{p_u} = \left[\sigma^2_{\mathbf{u}_1^1} \ldots \sigma^2_{\mathbf{u}_1^r} \ldots \sigma^2_{\mathbf{u}_N^1} \ldots \sigma^2_{\mathbf{u}_N^r}\right]^T$ actually corresponds to a power allocation for each of the data symbols so that $\sigma^2_{\mathbf{u}_i^k}$ represents the power assigned to the k-th symbol of the i-th user.

Now, let us introduce the matrix $\mathbf{W^2_{tx}}$ defined as:

$$\mathbf{W^2_{tx}} = \begin{pmatrix} \left|(\mathbf{W_{tx}})^{1,1}_{1,1}\right|^2 & \ldots & \left|(\mathbf{W_{tx}})^{1,r}_{1,1}\right|^2 & \ldots & \left|(\mathbf{W_{tx}})^{1,1}_{1,N}\right|^2 & \ldots & \left|(\mathbf{W_{tx}})^{1,r}_{1,N}\right|^2 \\ \vdots & \ddots & \vdots & \ldots & \vdots & \ddots & \vdots \\ \left|(\mathbf{W_{tx}})^{t,1}_{1,1}\right|^2 & \ldots & \left|(\mathbf{W_{tx}})^{t,r}_{1,1}\right|^2 & \ldots & \left|(\mathbf{W_{tx}})^{t,1}_{1,N}\right|^2 & \ldots & \left|(\mathbf{W_{tx}})^{t,r}_{1,N}\right|^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \left|(\mathbf{W_{tx}})^{1,1}_{M,1}\right|^2 & \ldots & \left|(\mathbf{W_{tx}})^{1,r}_{M,1}\right|^2 & \ldots & \left|(\mathbf{W_{tx}})^{1,1}_{M,N}\right|^2 & \ldots & \left|(\mathbf{W_{tx}})^{1,r}_{M,N}\right|^2 \\ \vdots & \ddots & \vdots & \ldots & \vdots & \ddots & \vdots \\ \left|(\mathbf{W_{tx}})^{t,1}_{M,1}\right|^2 & \ldots & \left|(\mathbf{W_{tx}})^{t,r}_{M,1}\right|^2 & \ldots & \left|(\mathbf{W_{tx}})^{t,1}_{M,N}\right|^2 & \ldots & \left|(\mathbf{W_{tx}})^{t,r}_{M,N}\right|^2 \end{pmatrix} \tag{2.22}$$

In other words, $\mathbf{W^2_{tx}}$ is obtained by element-wise application of the modulus squared operator $|\bullet|^2$ on the matrix $\mathbf{W_{tx}}$. Then, we can rewrite the previous equations as:

$$\mathbf{p} = \text{diag}(\mathbf{R_x}) = \mathbf{W^2_{tx}}\mathbf{p_u} \tag{2.23}$$

In that case, $\mathbf{p} \in \mathbb{R}^{Mt}$ contains the transmitted power in each antenna of the system. Hence, using the indexing scheme for vectors previously defined, we can rewrite the per-BTS transmitted power as:

$$P_{\text{tx,j}} = \sum_{l=1}^{t} \mathbf{p}_j^l = \left(\mathbf{A}\mathbf{W^2_{tx}}\mathbf{p_u}\right)_j \tag{2.24}$$

Where we have introduced:

$$\mathbf{A} = \mathbf{I}_{M\times M} \otimes \mathbf{1}_t^T \tag{2.25}$$

There $\otimes$ denotes the Kronecker product of two matrices and $\mathbf{1}_t$ the t-dimensional vector with all its entries having value 1. With $\mathbf{A}$ defined like that, the power consumed by the j-th BTS is simply given by the j-th element of $\mathbf{A}\mathbf{W}_{\mathbf{tx}}^{\mathbf{2}}\mathbf{p}_{\mathbf{u}}$ as shown in equation (2.24). This alternative form is useful since it allows to obtain the transmitted power (either per antenna or per BTS) as a linear function of the power allocation vector $\mathbf{p}_{\mathbf{u}}$, allowing for the usage of efficient numerical optimization algorithms based on convex programming.

2.1.3 Performance evaluation

The first parameter which we can use to evaluate the performance of a MIMO system is the autocorrelation matrix of the error vector, which is defined as:

$$\mathbf{R}_{\mathbf{e}} = \mathbb{E}\left\{(\mathbf{u}-\hat{\mathbf{u}})(\mathbf{u}-\hat{\mathbf{u}})^H\right\} \tag{2.26}$$

Substituting (2.2) and using the fact that noise and signal are uncorrelated we get:

$$\begin{aligned}\mathbf{R}_{\mathbf{e}} &= (\mathbf{I}-\mathbf{W}_{\mathbf{rx}}\mathbf{H}\mathbf{W}_{\mathbf{tx}})\,\mathbf{R}_{\mathbf{u}}\,(\mathbf{I}-\mathbf{W}_{\mathbf{rx}}\mathbf{H}\mathbf{W}_{\mathbf{tx}})^H + \mathbf{W}_{\mathbf{rx}}\mathbf{R}_{\mathbf{n}}\mathbf{W}_{\mathbf{rx}}^H = \\ &= \mathbf{W}_{\mathbf{rx}}\mathbf{H}\mathbf{W}_{\mathbf{tx}}\mathbf{R}_{\mathbf{u}}\mathbf{W}_{\mathbf{tx}}^H\mathbf{H}^H\mathbf{W}_{\mathbf{rx}}^H + \mathbf{R}_{\mathbf{u}} - \mathbf{W}_{\mathbf{rx}}\mathbf{H}\mathbf{W}_{\mathbf{tx}}\mathbf{R}_{\mathbf{u}} - \\ &- \mathbf{R}_{\mathbf{u}}\mathbf{W}_{\mathbf{tx}}^H\mathbf{H}^H\mathbf{W}_{\mathbf{rx}}^H + \mathbf{W}_{\mathbf{rx}}\mathbf{R}_{\mathbf{n}}\mathbf{W}_{\mathbf{rx}}^H\end{aligned} \tag{2.27}$$

The most direct interpretation of $\mathbf{R}_{\mathbf{e}}$ is that the trace of the $r \times r$ diagonal blocks, $(\mathbf{R}_{\mathbf{e}})_{i,i}$, contains the average error power between the set of symbols estimated by the i-th user, $\hat{\mathbf{u}}_i$, and the real set of symbols $\mathbf{u}_i$ that should have been ideally received. Such a magnitude is more usually referred to as Mean Squared Error (MSE). Most importantly, most of the cost functions employed to derive expressions for the linear transmit and receive filters can be formulated in some way as a function of $\mathbf{R}_{\mathbf{e}}$. Therefore, expression (2.27) will be of great importance for the developments which will follow along this project.

Another parameter, very related to the MSE, in the Signal to Noise plus Interference Ratio (SNIR). The name is, indeed, self-descriptive. This quantity measures the amount of signal power P_{s}, relative to the noise P_{n} and interference P_{int} powers. Mathematically:

$$SNIR = \frac{P_{\mathrm{s}}}{P_{\mathrm{n}} + P_{\mathrm{int}}} \tag{2.28}$$

The previous expression uses natural units (actually, the SNIR is a ratio of powers, thus it is dimensionless). However, it is most common to express it in decibels.

At this point, it will be useful to introduce the equivalent channel matrix, $\widetilde{\mathbf{H}}$, defined as:

$$\widetilde{\mathbf{H}} = \mathbf{W}_{\mathbf{rx}}\mathbf{H}\mathbf{W}_{\mathbf{tx}} \tag{2.29}$$

We will also introduce an equivalent noise vector $\mathbf{z}$ which represents the noise at the output of the receiver filter $\mathbf{W}_{\mathbf{rx}}$:

$$\mathbf{z} = \mathbf{W_{rx}}\mathbf{n} \tag{2.30}$$

Where $\mathbf{z}$ has autocovariance matrix $\mathbf{R_z} = \mathbf{W_{rx}}\mathbf{R_n}\mathbf{W_{rx}}^H$.

Depending on the expression of the filter matrix $\mathbf{W_{rx}}$, the noise vector $\mathbf{z}$ may be spatially correlated even when the original noise vector $\mathbf{n}$ is not. The reader is encouraged to note the analogy with the case when a time-domain filter colors the white noise at its input.

With this notation, the whole system breaks down to the extremely simple affine expression:

$$\hat{\mathbf{u}} = \widetilde{\mathbf{H}}\mathbf{u} + \mathbf{z} \tag{2.31}$$

Going back to the system model at "user level", we can expand the matrix $\widetilde{\mathbf{H}}$ as:

$$\widetilde{\mathbf{H}} = \begin{pmatrix} (\widetilde{\mathbf{H}})_{1,1} & (\widetilde{\mathbf{H}})_{1,2} & \dots & (\widetilde{\mathbf{H}})_{1,N} \\ (\widetilde{\mathbf{H}})_{2,1} & (\widetilde{\mathbf{H}})_{2,2} & \dots & (\widetilde{\mathbf{H}})_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ (\widetilde{\mathbf{H}})_{N,1} & (\widetilde{\mathbf{H}})_{N,2} & \dots & (\widetilde{\mathbf{H}})_{N,N} \end{pmatrix} \tag{2.32}$$

This equivalent channel matrix can be interpreted as follows. $(\widetilde{\mathbf{H}})_i \in \mathbb{C}^{r \times Nr}$ is the end-to-end channel seen by the i-th user. On the contrary, $(\widetilde{\mathbf{H}})^j \in \mathbb{C}^{Nr \times r}$ represents the contribution of the r data-streams corresponding to the j-th user in the set of all estimated symbols in the system, $\hat{\mathbf{u}}$. Finally $(\widetilde{\mathbf{H}})_{i,j} \in \mathbb{C}^{r \times r}\ \forall\, i = 1, \dots, N\, j = 1, \dots, N$ is the equivalent channel between the set of r data-streams transmitted by the j-th base-station and the set of r estimated symbols computed by the i-th receiver.

Similarly, the equivalent noise vector can be partitioned in N pieces as follows:

$$\mathbf{z} = \begin{pmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_N \end{pmatrix} \tag{2.33}$$

Where $\mathbf{z}_i \in \mathbb{C}^r \quad \forall i = 1, \dots, N$ is the filtered noise in the i-th user's terminal.

Therefore, we can express the data symbol estimate in the i-th receiver as:

$$\hat{\mathbf{u}}_i = (\widetilde{\mathbf{H}})_i \mathbf{u} + \mathbf{z}_i \tag{2.34}$$

Expanding it even further, we can interpret the different contributions to $\hat{\mathbf{u}}_i$:

$$\hat{\mathbf{u}}_i = \underbrace{(\widetilde{\mathbf{H}})_{i,i}\mathbf{u}_i}_{\text{desired signal}} + \underbrace{\sum_{k=1, k\neq i}^{N} (\widetilde{\mathbf{H}})_{i,k}\mathbf{u}_k}_{\text{interference}} + \underbrace{\mathbf{z}_i}_{\text{noise}} \tag{2.35}$$

From the previous equation it is straightforward to obtain all the terms needed to compute the SNIR. We have that:

$$P_{\mathrm{s}} = \mathrm{Tr}\left((\widetilde{\mathbf{H}})_{i,i}\mathbf{R}_{\mathbf{u}_i}(\widetilde{\mathbf{H}})_{i,i}^H\right) \tag{2.36}$$

$$P_{\mathrm{n}} = \mathrm{Tr}(\mathbf{R}_{\mathbf{z}_i}) \tag{2.37}$$

$$P_{\mathrm{int}} = \sum_{k=1,k\neq i}^{N} \mathrm{Tr}((\widetilde{\mathbf{H}})_{i,k}\mathbf{R}_{\mathbf{u}_k}(\widetilde{\mathbf{H}})_{i,k}^H) = \mathrm{Tr}\left((\widetilde{\mathbf{H}})_{i}\mathbf{R}_{\mathbf{u}}(\widetilde{\mathbf{H}})_{i}^H\right) - \mathrm{Tr}\left((\widetilde{\mathbf{H}})_{i,i}\mathbf{R}_{\mathbf{u}_i}(\widetilde{\mathbf{H}})_{i,i}^H\right) \tag{2.38}$$

For obtaining P_{int} we have used the fact that, whenever $\mathbf{R}_{\mathbf{u}}$ is a block-diagonal matrix as in (2.11):

$$(\widetilde{\mathbf{H}})_{i}\mathbf{R}_{\mathbf{u}}(\widetilde{\mathbf{H}})_{i}^H = \sum_{k=1}^{N} (\widetilde{\mathbf{H}})_{i,k}\mathbf{R}_{\mathbf{u}_k}(\widetilde{\mathbf{H}})_{i,k}^H \tag{2.39}$$

Putting all together we can finally express the SNIR for the i-th user as:

$$\mathsf{SNIR}_i = \frac{\mathrm{Tr}\left((\widetilde{\mathbf{H}})_{i,i}\mathbf{R}_{\mathbf{u}_i}(\widetilde{\mathbf{H}})_{i,i}^H\right)}{\mathrm{Tr}\left(\mathbf{R}_{\mathbf{z}_i}\right) + \mathrm{Tr}\left((\widetilde{\mathbf{H}})_{i}\mathbf{R}_{\mathbf{u}}(\widetilde{\mathbf{H}})_{i}^H\right) - \mathrm{Tr}\left((\widetilde{\mathbf{H}})_{i,i}\mathbf{R}_{\mathbf{u}_i}(\widetilde{\mathbf{H}})_{i,i}^H\right)} \tag{2.40}$$

Just as we computed the SNIR per user, we can compute the SNIR per antenna. However, we won't go through the derivations since, just as it happened when considered a per-antenna power constraint, computing the SNIR per antenna is a particular case of the formulation for computing the SNIR per user. From our real scenario, we can build a virtual scenario where each user is split into r users with a single receive antenna, and compute the SNIR per virtual user. We must note however that, unlike it may seem at first, if we compute the received signal power per antenna in all the antennas of a given user, and add all those power up, the resulting power will be smaller than the one obtained by the formula $P_{\mathrm{s}} = \mathrm{Tr}\left((\widetilde{\mathbf{H}})_{i,i}\mathbf{R}_{\mathbf{u}_i}(\widetilde{\mathbf{H}})_{i,i}^H\right)$. This, which may seem counterintuitive, is due to the fact that, when computing the SNIR per user, we consider the power received at each antenna of the i-th user due to any data symbol of the i-th user to be desired signal power. On the other hand, when computing the SNIR per antenna, we consider only as desired signal power for the i-th antenna the one due to exactly the i-th data symbol. For instance, in an scenario with $r = 2$, when computing the SNIR per user we would compute the desired signal power for the i-th user as the power in antenna 1 of that user due to data symbols 1 and 2 of the the i-th user and the power in antenna 2 due to data symbols 1 and 2 of that same user. However, when computing the SNIR per antenna, the desired power for antenna 1 of the i-th user would be the one due to data symbol 1 of that user, and the desired power for antenna 2 of the i-th user would be the one due to data symbol 2 of that user. In this case, the power in antenna 1 due to the data symbol 2, and the power in antenna 2 due to the data symbol 1 are regarded as interference. The meaning behind this is that, when computing the SNIR at user level, we assume that the receiver of that user will carry on further

signal processing to cancel the interference between the signal received at each of the antennas belonging to that particular terminal. That's why we don't want to count that as interference but as signal power when computing the SNIR at user level.

The SNIR gives very insightful information for the engineer in order to troubleshoot the system. However, the ultimate measure of performance in any communications system has to be actually a quantification of the amount of information we can transfer using such a system. As we know, the magnitude which tries to quantify that is nothing but the mutual information, as defined by Claude Shannon in [7]. The mutual information for an AWGN vector channel of the canonical form:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \tag{2.41}$$

Can be shown to be [8]:

$$C = \log_2\left(\left|\mathbf{I}_{Nr} + \mathbf{H}\mathbf{R}_{\mathbf{x}}\mathbf{H}^H\mathbf{R}_{\mathbf{n}}^{-1}\right|\right) \tag{2.42}$$

Using (2.31) we can find the overall rate of the whole MIMO system,including the precoder and the receive filter, to be:

$$R = \log_2\left(\left|\mathbf{I}_{Nr} + \widetilde{\mathbf{H}}\mathbf{R}_{\mathbf{u}}\widetilde{\mathbf{H}}^H\mathbf{R}_{\mathbf{z}}^{-1}\right|\right) \tag{2.43}$$

Much more interesting is to know the rate for the i-th user, which can be approximated by treating the interference as noise, so that the equivalent noise covariance matrix would become:

$$\mathbf{R}_{\mathbf{n}\text{eq}} = \mathbf{R}_{\mathbf{z}_i} + \sum_{k=1,k\neq i}^{N} (\widetilde{\mathbf{H}})_{i,k}\mathbf{R}_{\mathbf{u}_k}(\widetilde{\mathbf{H}})_{i,k}^H \tag{2.44}$$

Then:

$$R_i = \log_2\left(\left|\mathbf{I}_r + (\widetilde{\mathbf{H}})_{i,i}\mathbf{R}_{\mathbf{u}}(\widetilde{\mathbf{H}})_{i,i}^H\left(\mathbf{R}_{\mathbf{z}_i} + \sum_{k=1,k\neq i}^{N}(\widetilde{\mathbf{H}})_{i,k}\mathbf{R}_{\mathbf{u}_k}(\widetilde{\mathbf{H}})_i^H\right)^{-1}\right|\right) \tag{2.45}$$

The reader must note that the previous equations hold only for a single channel realization. However, the channel matrix $\mathbf{H}$ and, thus, $\widetilde{\mathbf{H}}$ are stochastic matrices. In order to properly compute the rate we must average the previous expression over the probability density function of $\mathbf{H}$ as:

$$R = \mathbb{E}\left\{\log_2\left(\left|\mathbf{I}_{Nr} + \widetilde{\mathbf{H}}\mathbf{R}_{\mathbf{u}}\widetilde{\mathbf{H}}^H\mathbf{R}_{\mathbf{z}}^{-1}\right|\right)\right\} \tag{2.46}$$

$$R_i = \mathbb{E}\left\{\log_2\left(\left|\mathbf{I}_r + (\widetilde{\mathbf{H}})_{i,i}\mathbf{R}_{\mathbf{u}}(\widetilde{\mathbf{H}})_{i,i}^H\left(\mathbf{R}_{\mathbf{z}_i} + \sum_{k=1,k\neq i}^{N}(\widetilde{\mathbf{H}})_{i,k}\mathbf{R}_{\mathbf{u}_k}(\widetilde{\mathbf{H}})_i^H\right)^{-1}\right|\right)\right\} \tag{2.47}$$

2.1.4 Relation to OFDM systems

There exists a very strong bond between the formulations carried on during this project and the design of OFDM system. We started by stating that we would consider only systems without ISI, which may sound as a too optimistic hypothesis. However, state-of-the-art communication systems are prone to employ OFDM techniques in one way or another. Probably, the maximum exponent of this trend is the Long Term Evolution (LTE) 3GPP standard for the forth generation of mobile communications (4G).

OFDM divides the bandwidth into a set of N_c disjoint, narrow-band carriers. The carriers are mutually orthogonal (hence the name of Orthogonal Frequency Division Multiplex). As a consequence, we can consider that each of the carriers is completely independent of the others and study the system on a per-carrier basis. Moreover, since each of the carriers occupy a very narrow frequency band, each of the N_c parallel subchannels which conform the whole OFDM system will experience flat and slow fading so that the channel model we have been employing describes the per-carrier behavior of an OFDM system with great accuracy.

Mathematically, we can model a MIMO OFDM system as:

$$\hat{\mathbf{u}}[k] = \mathbf{W_{rx}}[k]\mathbf{H}[k]\mathbf{W_{tx}}[k]\mathbf{u}[k] + \mathbf{W_{rx}}[k]\mathbf{n}[k] \quad \forall\, k = 1, \ldots, N_c \tag{2.48}$$

Note that the notation $[k]$ refers to the fact that we have N_c independent OFDM carriers, each giving rise to a different parallel MIMO system. In other words, we are simply applying equation (2.2) N_c times in a completely independent fashion. To find the precoders and receiver, we simply must follow the procedures described in this project N_c times independently. Not only does this greatly reduce computational complexity, but also allows the usage of multi-core or even distributed computational environments to greatly speed-up the computations required for filter design. However, recently, some studies have proposed optimization algorithms which operate jointly on the set of carriers to find the precoder matrix and receive filters which have been shown to outperform more traditional designs. Therefore, the relation between traditional design of MIMO filters and OFDM systems remains an open problem with good perspectives for further studies.

Given the focus on this project on mobile communications and, more precisely, 4G technology, all the derivations done here are actually thought to be employed within an inherent OFDM system as described in this subsection. However, from now on, we will omit the usage of the superscript k in order to keep notation as simple as possible. The reader can then infer that any procedures shown here would actually be performed on a per-carrier basis N_c times.

Chapter 3

MIMO linear filter design

3.1 Introduction to MIMO linear filter and precoder design

In chapter 2, we defined with sufficient detail the system we are going to work with. Now we will proceed to one of the most important sections of this project, where we will expose some state-of-the-art algorithms and formulations to design the precoder $\mathbf{W_{tx}}$ and the receiver $\mathbf{W_{rx}}$ to finally push the current designs a bit forward by introducing some novel filter designs specially engineered to tackle the particularities of the mobile communications scenario we are dealing with along this project.

As we have shown in equation (2.35), the symbol data vector estimate computed by the i-th receiver is nothing but a term containing the desired signal $\mathbf{u}_i$, more precisely, a linear function of that desired signal $(\widetilde{\mathbf{H}})_{i,i}\mathbf{u}_i$, corrupted by interference and spatially colored noise. According to that observation, the different filter designs shown in the literature can be roughly classified as:

Matched filters: Focus only on the noise by trying to maximize the SNR (not the SNIR!).

Zero forcing filters: Focus only on the interference by trying to cancel it.

MMSE filters: Try to achieve a compromise between noise reduction and interference cancellation.

Of course, such a classification is by no means complete. Lots of different filtering schemes have been proposed in many articles and today many more continue to be engineered. Some of them try to improve the basic, well-known filter designs by combining several of the main types or adding slight modifications in the formulations.

Moreover, it is really important at this point to consider yet another coupling issue: the design of $\mathbf{W_{rx}}$ and $\mathbf{W_{tx}}$ is, in general, linked. To realize why that happens, we just need to make the following observation. Obviously, for the design of both matrices, no matter which filtering paradigm or formulation we choose, the channel matrix $\mathbf{H}$ will strongly influence the

solution. However, from the point of view of the design of the receive filter $\mathbf{W_{rx}}$, the effective channel matrix is not $\mathbf{H}$ but the chain $\mathbf{HW_{tx}}$. Similarly, for designing $\mathbf{W_{tx}}$, there is no way of separating $\mathbf{W_{rx}}$ from the design as the effective channel matrix becomes $\mathbf{W_{rx}H}$.

Sadly, the joint design of the precoder and the receive filter is a mathematical problem with an enormous complexity, both analytical and computational. Because of that, it is a problem whose solution has not been found yet. The vast majority of the present schemes, focus either on a precoding design, where the formulation tries to find the optimal $\mathbf{W_{tx}}$ and employ no receive filter or a receive filtering based design, where we optimize $\mathbf{W_{rx}}$ with no precoding. The first case is usually employed for the downlink channel whereas the latter is most commonly used in the uplink, simply because it is better to place the complexity of the signal processing in the BTS part than in the user part. As we shall see in the following subsections, there are some designs which achieve a suboptimal joint design. For instance, in [9], they design a precoding scheme which finds the optimal $\mathbf{W_{tx}}$ under a TPC while simultaneously finding a simplified $\mathbf{W_{rx}}$ of the form $\mathbf{W_{rx}} = \alpha\mathbf{I}$, where only the scalar α is optimized in the receive part. Indeed, some of our own contributions in this project are actually directed towards increasing the number of degrees of freedom for the design of $\mathbf{W_{rx}}$ with respect to the work in [9].

It is also important to note that, only for the single-user scenario, the problem for the joint design of $\mathbf{W_{tx}}$ and $\mathbf{W_{rx}}$ was brilliantly solved in [10] by employing a mathematical framework based on vector majorization.

In the subsequence subsections, we will explore in greater detail each of the most relevant options for the design of precoding matrices $\mathbf{W_{tx}}$ and receive filter $\mathbf{W_{rx}}$. In almost all the cases, the mathematical formulation to reach an expression for $\mathbf{W_{tx}}$ or $\mathbf{W_{rx}}$ will be based on optimization theory.

In some cases, it will be an unconstrained optimization problem, where we can simply try to find the value for the variable to be optimized that makes the first derivate of the cost function vanish. Being more general, we should check that the singular point we just found is indeed a minimum or a maximum, depending on what we want. However, most of the cost functions we will minimize will be convex functions and, conversely, most of the cost functions we will maximize will be concave functions. Therefore, we can guarantee that any singular point will be an optimum for our problem.

However, in other cases we will have to solve a constrained optimization problem, where a more complex mathematical theory for optimization needs to be used. The reader can find a comprehensive study of those techniques (Lagrange multipliers, convex optimization theory, KKT conditions, etc.) in [11] or [12]. A lighter introduction to the topic can be found as an appendix in [13].

We must also note that we will working with cost functions which are actually real-valued functions of several complex-valued independent variables. Those functions pose a very big difficulty: they are not analytic and, therefore, not differentiable in the usual sense. However, in [14] the reader can find a mathematical workaround which allows to redefine the differentiation

process for non-analytic complex functions in such a way that it is useful for optimization purposes. That is, singular points actually correspond to minima/maxima/saddle points of the real-valued functions.

3.2 Matched filters

The first kind of filters we are going to study are denoted as matched filters. MIMO matched filters are in complete analogy with time-domain matched filters: they are the result of a mathematical optimization process which tries to find the expression for the filter which maximizes the Signal-To-Noise Ratio (SNR). Mathematically, the SNR is defined as:

$$\mathsf{SNR} = \frac{P_{\mathrm{s}}}{P_{\mathrm{n}}} \tag{3.1}$$

In that equation, P_{n} is simply the total noise power at the output of the receiver, $P_{\mathrm{n}} = \mathrm{Tr}(\mathbf{W_{rx}} \mathbf{R_n} \mathbf{W_{rx}}^H)$. On the other hand, the definition of the signal power P_{s} is trickier, since we need to define what is (desired) signal power, and what is interference power. Indeed, one definition or another would lead to different designs. In our case, we will follow the work in [9] and define the desired signal power received by the i-th user as the power of the projection of the estimated symbol vector $\hat{\mathbf{u}}_i$ on the original symbol vector $\mathbf{u}_i$. Projections in that sense are to be understood under the Hilbert space defined by the set of random variables we are working with. Hence, the desired signal power at the i-th user's receiver is equivalent to the squared cross-correlation between $\hat{\mathbf{u}}_i$ and $\mathbf{u}_i$. Mathematically:

$$P_{\mathrm{s,i}} = \left| \mathbb{E}\left\{ \mathbf{u}_i^H \hat{\mathbf{u}}_i \right\} \right|^2 \tag{3.2}$$

And the total desired signal power is computed simply as the sum of the desired signal power for each user:

$$P_{\mathrm{s}} = \sum_{i=1}^{N} P_{\mathrm{s,i}} = \sum_{i=1}^{N} \left| \mathbb{E}\left\{ \mathbf{u}_i^H \hat{\mathbf{u}}_i \right\} \right|^2 = \left| \mathbb{E}\left\{ \mathbf{u}^H \hat{\mathbf{u}} \right\} \right|^2 \tag{3.3}$$

The criteria of maximizing the SNR is very typical in communication systems and even other kind of related systems, like RADAR. As a consequence, matched-filters are a common subject of study in undergraduate and graduate courses of electrical engineering, making them widely-known. Nonetheless, we must note that, for this project, we are including them more for completeness' sake than for their utility in this particular scenario. Mobile communication systems are very rarely limited by noise, being the interference the main source of trouble for engineers. However, the formulation of a matched filter tries to maximize the SNR, not the SNIR. In other words, these filters ignore interference, which is actually our main concern in this kind of communication systems. As a consequence, the performance of MIMO matched-filters in wireless cellular systems tends to be poor.

3.2.1 Tx-MF

We are going to present a matched-filter based precoder which assumes full coordination in the transmitter, none in the receiver and a TPC. Therefore, it can be employed for the single-user scenarios, both uplink and downlink, and also in multi-user scenarios only for the downlink channel. Under this setting, $\mathbf{W_{rx}}$ will be regarded as constant and, therefore, the noise power P_{n} will be constant too. As a consequence, maximizing the SNR is fully equivalent in this case to maximizing P_{s}, since the variable we are optimizing, $\mathbf{W_{tx}}$, cannot change the value of P_{n}. Taking that into account, we can formulate the computation of the Tx-MF precoder as the following constrained optimization problem:

$$\max_{\mathbf{W_{tx}}} \left|\mathbb{E}\left\{\mathbf{u}^H\hat{\mathbf{u}}\right\}\right|^2 \quad s.t. \quad \mathbb{E}\left\{\|\mathbf{W_{tx}u}\|^2\right\} \leq P_{\max} \tag{3.4}$$

Note that we have included a TPC as it was discussed in section 2.1.2. Developing the terms shown in the previous equation we have that:

$$\left|\mathbb{E}\left\{\mathbf{u}^H\hat{\mathbf{u}}\right\}\right|^2 = \mathbb{E}\left\{\hat{\mathbf{u}}^H\mathbf{u}\right\}\mathbb{E}\left\{\mathbf{u}^H\hat{\mathbf{u}}\right\} = \mathrm{Tr}\left(\mathbf{W_{rx}HW_{tx}R_u}\right)\mathrm{Tr}\left(\mathbf{W_{tx}^H H^H W_{rx}^H R_u}\right) \tag{3.5}$$

And:

$$\mathbb{E}\left\{\|\mathbf{W_{tx}u}\|^2\right\} = \mathrm{Tr}\left(\mathbf{W_{tx}R_u W_{tx}^H}\right) \tag{3.6}$$

Therefore, we can write the Lagrangian for the constrained optimization problem as

$$\begin{aligned}\mathcal{L}(\mathbf{W_{tx}},\lambda) = &-\mathrm{Tr}\left(\mathbf{W_{rx}HW_{tx}R_u}\right)\mathrm{Tr}\left(\mathbf{W_{tx}^H H^H W_{rx}^H R_u}\right)\\ &+\lambda\left(\mathrm{Tr}\left(\mathbf{W_{tx}R_u W_{tx}^H}\right) - P_{\max}\right)\end{aligned} \tag{3.7}$$

Taking the derivative with respect to $\mathbf{W_{tx}^H}$ we can get:

$$\frac{\partial\mathcal{L}(\mathbf{W_{tx}},\lambda)}{\partial\mathbf{W_{tx}^H}} = \lambda\mathbf{W_{tx}R_u} - \mathrm{Tr}\left(\mathbf{W_{rx}HW_{tx}R_u}\right)\mathbf{H}^H\mathbf{W_{rx}^H R_u} \tag{3.8}$$

Equating the derivative to 0 and solving for $\mathbf{W_{tx}}$ we find:

$$\mathbf{W_{tx}} = \frac{1}{\lambda}\mathrm{Tr}\left(\mathbf{W_{rx}HW_{tx}R_u}\right)\left(\mathbf{W_{rx}H}\right)^H = \alpha\left(\mathbf{W_{rx}H}\right)^H \tag{3.9}$$

Clearly, the function to be maximized increases monotonically with α as α^2. Hence, the optimal choice for α is the biggest value which still satisfies the TPC. By substituting the expression we just found for $\mathbf{W_{tx}}$ into the TPC:

$$\begin{aligned}P_{\max} &= \mathrm{Tr}\left(\mathbf{W_{tx}R_u W_{tx}^H}\right) = \alpha^2\,\mathrm{Tr}\left(\left(\mathbf{W_{rx}H}\right)^H\mathbf{R_u}\left(\mathbf{W_{rx}H}\right)\right)\\ \alpha &= \sqrt{\frac{P_{\max}}{\mathrm{Tr}\left(\left(\mathbf{W_{rx}H}\right)^H\mathbf{R_u}\left(\mathbf{W_{rx}H}\right)\right)}}\end{aligned} \tag{3.10}$$

So that the final expression for the Tx-MF precoding matrix becomes:

$$\mathbf{W_{tx}} = \sqrt{\frac{P_{\max}}{\mathrm{Tr}\left(\left(\mathbf{W_{rx}H}\right)^H \mathbf{R_u} \left(\mathbf{W_{rx}H}\right)\right)}} \left(\mathbf{W_{rx}H}\right)^H \tag{3.11}$$

3.2.2 Rx-MF

Now we will study the dual filter of the Tx-MF precoder: the Rx-MF receive filter. In this case, we will assume full coordination in the receiver and none in the transmitter. Moreover, since we are going to optimize only over $\mathbf{W_{rx}}$, with $\mathbf{W_{tx}}$ supposed a known constant, we don't need to consider any power constraint. Therefore, this filter can be used for single-user scenarios, both uplink and downlink, and multi-user scenarios only for the uplink channel. Even though we can drop the power constraint and, thus, the formulation is simpler as it is an unconstrained optimization problem, now we must realize that the noise power P_{n} is no longer constant and we have to optimize a cost function which is a quotient of traces. The following unconstrained optimization problem can be used to obtain the expression for the Rx-MF:

$$\max_{\mathbf{W_{rx}}} \frac{\left|\mathbb{E}\left\{\mathbf{u}^H\hat{\mathbf{u}}\right\}\right|^2}{\mathbb{E}\left\{\left\|\mathbf{W_{rx}n}\right\|^2\right\}} \tag{3.12}$$

Just like before, we have that:

$$\left|\mathbb{E}\left\{\mathbf{u}^H\hat{\mathbf{u}}\right\}\right|^2 = \mathbb{E}\left\{\hat{\mathbf{u}}^H\mathbf{u}\right\}\mathbb{E}\left\{\mathbf{u}^H\hat{\mathbf{u}}\right\} = \mathrm{Tr}\left(\mathbf{W_{rx}HW_{tx}R_u}\right)\mathrm{Tr}\left(\mathbf{W_{tx}^H H^H W_{rx}^H R_u}\right) \tag{3.13}$$

We noise power is simply:

$$\mathbb{E}\left\{\left\|\mathbf{W_{rx}n}\right\|^2\right\} = \mathrm{Tr}\left(\mathbf{W_{rx}R_nW_{rx}^H}\right) \tag{3.14}$$

Using the last two equations, we can see that optimizing the SNR amounts to finding the maximum of the following function:

$$f(\mathbf{W_{rx}}) = \frac{\mathrm{Tr}\left(\mathbf{W_{rx}HW_{tx}R_u}\right)\mathrm{Tr}\left(\mathbf{W_{tx}^H H^H W_{rx}^H R_u}\right)}{\mathrm{Tr}\left(\mathbf{W_{rx}R_nW_{rx}^H}\right)} \tag{3.15}$$

For this, we first compute the derivative of $f(\mathbf{W_{rx}})$ to find:

$$\frac{\partial f(\mathbf{W_{rx}})}{\partial \mathbf{W_{rx}^H}} = \mathrm{Tr}\left(\mathbf{W_{rx}HW_{tx}R_u}\right)\frac{\mathrm{Tr}\left(\mathbf{W_{rx}R_nW_{rx}^H}\right)\mathbf{R_uW_{tx}^H H^H} - \mathrm{Tr}\left(\mathbf{W_{tx}^H H^H W_{rx}^H R_u}\right)\mathbf{W_{rx}R_n}}{\left(\mathrm{Tr}\left(\mathbf{W_{rx}R_nW_{rx}^H}\right)\right)^2} \tag{3.16}$$

Equating the right hand side to zero and solving for $\mathbf{W_{rx}}$ we get:

$$\mathbf{W_{rx}} = \frac{\mathrm{Tr}\left(\mathbf{W_{rx}R_nW_{rx}^H}\right)}{\mathrm{Tr}\left(\mathbf{W_{tx}^H H^H W_{rx}^H R_u}\right)}\mathbf{R_u}\left(\mathbf{HW_{tx}}\right)^H\mathbf{R_n^{-1}} = \alpha\mathbf{R_u}\left(\mathbf{HW_{tx}}\right)^H\mathbf{R_n^{-1}} \tag{3.17}$$

We can straightforwardly check that the value of α is irrelevant since both numerator and denominator of the ratio (3.12) to be maximized depend on α as α^2. Therefore, for simplicity, we assume that $\alpha = 1$, dropping the term.

The final expression for the Rx-MF receive filter is then:

$$\mathbf{W_{rx}} = \mathbf{R_u} \left(\mathbf{H W_{tx}}\right)^H \mathbf{R_n}^{-1} \tag{3.18}$$

3.3 Zero Forcing filters

Zero-forcing filters can be considered to be the opposite concept to that of matched-filters. Whereas matched filters cared only about noise, trying to maximize the SNR but forgetting about the interference, zero-forcing filters care only about interference, trying to completely cancel it (hence the name of zero-forcing i.e. it forces the interference to be zero). However, as the reader may guess, the price to pay for completely canceling the interference is that the SNR will decrease.

In scenarios where the interference strongly dominates over the noise, decrease the SNR at the expense of completely eliminating the interference is a really sensible design. Because of this, zero-forcing filters are widely used in state-of-the-art MIMO systems and continue to be included into standards forecast for the next decade.

Another very interesting property of zero-forcing designs, which actually is a direct consequence of forcing the interference to be zero, is that the equivalent channel matrix $\widetilde{\mathbf{H}}$ becomes a block-diagonal matrix:

$$\widetilde{\mathbf{H}} = \begin{pmatrix} (\widetilde{\mathbf{H}})_{1,1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & (\widetilde{\mathbf{H}})_{2,2} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & (\widetilde{\mathbf{H}})_{N,N} \end{pmatrix} \tag{3.19}$$

In this case, the model can be break-down on a per-user basis, since $\widetilde{\mathbf{H}}$ being block-diagonal means that we have a set of N parallel, non-interacting subchannels. The end-to-end subchannel corresponding to the i-th user becomes:

$$\hat{\mathbf{u}}_i = (\widetilde{\mathbf{H}})_{i,i}\mathbf{u}_i + \mathbf{z}_i \quad \forall\, i = 1, \dots, N \tag{3.20}$$

Zero-Forcing filters which construct an equivalent channel such as the one in equation (3.20) are commonly referred to as Block-Diagonalization (BD) methods.

3.3.1 Tx-ZF

We will start our study of Zero Forcing schemes by learning how to design a basic ZF precoder. We will follow the scheme proposed in [9] and obtain a precoder which uses full coordination in

the transmitter and no coordination in the receiver, with a TPC. Subsequently, this criterion is useful for the single-user scenario and the downlink channel of the multi-user case.

The idea is actually really simple: we want a fully diagonalized channel where the signal received by the i-th user is of the form:

$$\mathbf{y}_i = \alpha \mathbf{u}_i + \mathbf{z}_i \tag{3.21}$$

That is, each user receives just the data symbols corresponding to that user, possibly attenuated, plus some noise. The interference between users is zero since the signal received by the i-th user does not depend on $\mathbf{u}_j$ for $j \neq i$. Not only that, in this scheme we are also requiring that the signal received by the k-th antenna of the i-th user, $\mathbf{y}_i^k$, depends only on the symbol sent on the k-th data stream of user i, $\mathbf{u}_i^k$. That is, the interference has to be forced to zero not only at user level, but also at antenna level. Finally, we are also asking that the attenuation suffered by the signal, α, is constant for all users, and for all the antennas of each user too. Then, we can see that this is fully equivalent to say that the equivalent channel matrix, $\widetilde{\mathbf{H}} = \mathbf{W_{rx}} \mathbf{H} \mathbf{W_{tx}}$ has to be the identity matrix of size $Nr \times Nr$.

By definition, a matrix which satisfies that is the Moore-Penrose pseudo-inverse of $\mathbf{W_{rx}}\mathbf{H}$, that is, the matrix $(\mathbf{W_{rx}}\mathbf{H})^+$. However, we will follow the derivations in [9], where they use instead a constrained optimization problem which tries to minimize the total transmit power, under the constraint $\widetilde{\mathbf{H}} = \mathbf{I}$. As we will see, the TPC will be enforced later by the introduction of a scalar factor.

Mathematically, the optimization problem can be written as:

$$\min_{\mathbf{W_{tx}}} \mathbb{E}\left\{ \|\mathbf{W_{tx}}\mathbf{u}\|^2 \right\} \quad s.t. \quad \mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}} = \mathbf{I} \tag{3.22}$$

As we studied in section 2.1.2, the total transmitted power is:

$$P_{\text{tx}} = \mathbb{E}\left\{ \|\mathbf{W_{tx}}\mathbf{u}\|^2 \right\} = \text{Tr}\left(\mathbf{W_{tx}} \mathbf{R_u} \mathbf{W_{tx}}^H \right) \tag{3.23}$$

Using that, we can write the Lagrangian for this optimization problem as:

$$\mathcal{L}\left(\mathbf{W_{tx}}, \mathbf{\Phi}\right) = \text{Tr}\left(\mathbf{W_{tx}} \mathbf{R_u} \mathbf{W_{tx}}^H \right) - \text{Tr}\left(\left(\mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}} - \mathbf{I} \right) \mathbf{\Phi}^H \right) \tag{3.24}$$

Where we have introduced:

$$\mathbf{\Phi} = \begin{pmatrix} \phi_{1,1} & \cdots & \phi_{1,Nr} \\ \vdots & \ddots & \vdots \\ \phi_{Nr,1} & \cdots & \phi_{Nr,Nr} \end{pmatrix} \tag{3.25}$$

A matrix of dual variables associated to the $(Nr)^2$ equality constraints imposed by the condition $\mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}} = \mathbf{I}$. Even if it may not be obvious at first sight, we can find the term $\text{Tr}((\mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}} - \mathbf{I})\mathbf{\Phi}^H)$ to be completely equivalent to the typical summation term added for

constructing a Lagrangian in optimization problems with equality constraints. Convex optimization books usually add a term of the form $\sum_{i=1}^{N} v_i h_i(\mathbf{x})$, where $\{v_i\}_{i=1}^{N}$ are the Lagrange multipliers, and $\{h_i(\mathbf{x})\}_{i=1}^{N}$ are the set of equality constraints so that a point $\mathbf{x}$ is feasible if and only if $h_i(\mathbf{x}) = 0$ for all i. However, if we develop the matrix product $(\mathbf{W_{rx}HW_{tx}} - \mathbf{I})\,\mathbf{\Phi}^H$ and compute the trace, we see that it is a summation of the same kind, where each of the $(Nr)^2$ $\phi_{i,j}$ take the role of one of the v_i, and every of the $(Nr)^2$ elements of the matrix $\mathbf{W_{rx}HW_{tx}} - \mathbf{I}$ is an equality constraint $h_i(\mathbf{W_{tx}})$ that should be satisfied.

Differentiating the Lagrangian and equating it to zero we get:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W_{tx}}} = \mathbf{R_u W_{tx}^H} - \mathbf{\Phi}^H \mathbf{W_{rx} H} = 0 \tag{3.26}$$

Solving for $\mathbf{W_{tx}}$:

$$\mathbf{W_{tx}} = \mathbf{H}^H \mathbf{W_{rx}^H} \mathbf{\Phi} \mathbf{R_u^{-1}} \tag{3.27}$$

Substituting that into the constraint $\mathbf{W_{rx}HW_{tx}} = \mathbf{I}$ we can write:

$$\mathbf{W_{rx}} \mathbf{H} \mathbf{H}^H \mathbf{W_{rx}^H} \mathbf{\Phi} \mathbf{R_u^{-1}} = \mathbf{I} \tag{3.28}$$

So that we get:

$$\mathbf{\Phi} = \left(\mathbf{W_{rx}} \mathbf{H} \mathbf{H}^H \mathbf{W_{rx}^H}\right)^{-1} \mathbf{R_u} \tag{3.29}$$

The expression for the precoder matrix is then:

$$\mathbf{W_{tx}} = \mathbf{H}^H \mathbf{W_{rx}^H} \left(\mathbf{W_{rx}} \mathbf{H} \mathbf{H}^H \mathbf{W_{rx}^H}\right)^{-1} \tag{3.30}$$

However, we must still take into account the TPC $\mathbb{E}\left\{\|\mathbf{W_{tx}u}\|^2\right\} \leq P_{\max}$. The optimization problem we are trying to solve tries to minimize the total transmitted power while keeping the interference between data streams null. However, there is no guarantee at all that the minimum we reach will be below the TPC, $P_{\max}$. To fix that issue, in [9], they take a heuristic approximation consisting on normalizing the previous expression for the precoder $\mathbf{W_{tx}}$ by multiplying it by a scalar γ. Doing that, the final expression for the Tx-ZF precoder matrix is:

$$\mathbf{W_{tx}} = \gamma \mathbf{H}^H \mathbf{W_{rx}^H} \left(\mathbf{W_{rx}} \mathbf{H} \mathbf{H}^H \mathbf{W_{rx}^H}\right)^{-1} \tag{3.31}$$

Where the value of γ is such that the constraint $\mathbb{E}\left\{\|\mathbf{W_{tx}u}\|^2\right\} \leq P_{\max}$ is satisfied. Developing the expression:

$$\begin{aligned}\mathbb{E}\left\{\|\mathbf{W_{tx}u}\|^2\right\} &= \mathrm{Tr}\left(\mathbf{W_{tx}R_u W_{tx}}^H\right)\\ \mathrm{Tr}\left(\mathbf{W_{tx}R_u W_{tx}}^H\right) &= \gamma^2\,\mathrm{Tr}\left(\left(\mathbf{W_{rx}HH}^H\mathbf{W_{rx}}^H\right)^{-1}\mathbf{R_u}\right)\\ \gamma &= \sqrt{\frac{P_{\max}}{\mathrm{Tr}\left(\left(\mathbf{W_{rx}HH}^H\mathbf{W_{rx}}^H\right)^{-1}\mathbf{R_u}\right)}}\end{aligned} \tag{3.32}$$

This heuristic approximation can be enhanced if we introduce the TPC directly into the optimization problem and increase the number of degrees of freedom in the system by taking into account that the constraint $\mathbf{W_{rx}HW_{tx}} = \mathbf{I}$ is not strictly necessary to remove the interference. Indeed, having $\mathbf{W_{rx}HW_{tx}} = \mathbf{B}$ with $\mathbf{B}$ an arbitrary diagonal matrix, we get no interference between data streams. This observation gives rise to the first of the original contributions developed along this project in the design of MIMO linear precoders. That new precoding scheme, which we will call Generalized Tx-ZF, can be found in section 3.3.3.

3.3.2 Rx-ZF

Once we have seen how to design a basic ZF precoder, we will turn our attention to the dual case and design a basic ZF receive filter. As we saw in the formulation of the Rx-MF, since we will be optimizing only over $\mathbf{W_{rx}}$, the transmitted power will be irrelevant in the formulation and we will employ no transmitted power constraint of any kind. Indeed, this will happen in the formulation of every receive filter unless we use a joint optimization of $\mathbf{W_{tx}}$ and $\mathbf{W_{rx}}$.

We will be assuming full coordination in the receiver and no coordination in the transmitter, hence the Rx-ZF criterion is to be used in single-user scenarios and in the uplink channel of multi-user scenarios only.

For designing the receive filter $\mathbf{W_{rx}}$ according to the Rx-ZF scheme, we will assume $\mathbf{W_{tx}}$ fixed and known. The main target is to obtain $\mathbf{W_{rx}HW_{tx}} = \mathbf{I}$. Again, the Moore-Penrose pseudo-inverse of $\mathbf{HW_{tx}}$ is a valid solution for the problem. Nevertheless, just like for the Tx-ZF,we will use the derivation proposed in [9]. In that paper, the following constrained optimization problem is formulated:

$$\min_{\mathbf{W_{rx}}} \mathbb{E}\left\{\|\mathbf{u}-\hat{\mathbf{u}}\|^2\right\} \quad s.t. \quad \mathbf{W_{rx}HW_{tx}} = \mathbf{I} \tag{3.33}$$

In words, in [9] they formulate the calculation of the Rx-ZF filter matrix as trying to minimize the MSE under the restriction that the equivalent channel matrix $\widetilde{\mathbf{H}} = \mathbf{W_{rx}HW_{tx}}$ is the identity matrix. Substituting $\mathbf{W_{rx}HW_{tx}} = \mathbf{I}$ in (2.27) we obtain:

$$\mathbf{R_e} = \mathbf{W_{rx}R_n}\mathbf{W_{rx}}^H \tag{3.34}$$

Hence, the MSE simply becomes:

$$\mathbb{E}\left\{\|\mathbf{u}-\hat{\mathbf{u}}\|^2\right\} = \mathrm{Tr}\left(\mathbf{R_e}\right) = \mathrm{Tr}\left(\mathbf{W_{rx}}\mathbf{R_n}\mathbf{W_{rx}}^H\right) \tag{3.35}$$

The optimization problem can then be written as:

$$\min_{\mathbf{W_{rx}}} \mathrm{Tr}\left(\mathbf{W_{rx}}\mathbf{R_n}\mathbf{W_{rx}}^H\right) \quad s.t. \quad \mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}} = \mathbf{I} \tag{3.36}$$

And the Lagrangian associated to the constrained optimization problem is:

$$\mathcal{L}(\mathbf{W_{rx}}, \mathbf{\Phi}) = \mathrm{Tr}\left(\mathbf{W_{rx}}\mathbf{R_n}\mathbf{W_{rx}}^H\right) - \mathrm{Tr}\left(\left(\mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}} - \mathbf{I}\right)\mathbf{\Phi}^H\right) \tag{3.37}$$

Differentiating the Lagrangian with respect to $\mathbf{W_{rx}}$ we obtain:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W_{rx}}} = \mathbf{R_n}\mathbf{W_{rx}}^H - \mathbf{H}\mathbf{W_{tx}}\mathbf{\Phi}^H \tag{3.38}$$

Equating $\frac{\partial \mathcal{L}}{\partial \mathbf{W_{rx}}} = 0$ and solving for $\mathbf{W_{rx}}$ we get:

$$\mathbf{W_{rx}} = \mathbf{\Phi}\mathbf{W_{tx}}^H\mathbf{H}^H\mathbf{R_n}^{-1} \tag{3.39}$$

Substituting that result in the constraint $\mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}} = \mathbf{I}$ we get:

$$\mathbf{\Phi}\mathbf{W_{tx}}^H\mathbf{H}^H\mathbf{R_n}^{-1}\mathbf{H}\mathbf{W_{tx}} = \mathbf{I} \tag{3.40}$$

We can then observe that the matrix $\mathbf{W_{tx}}^H\mathbf{H}^H\mathbf{R_n}^{-1}\mathbf{H}\mathbf{W_{tx}}$ will be a square matrix no matter how many transmit and receive antennas we have in the system. Assuming that the previous matrix is non-singular, we can write:

$$\mathbf{\Phi} = (\mathbf{W_{tx}}^H\mathbf{H}^H\mathbf{R_n}^{-1}\mathbf{H}\mathbf{W_{tx}})^{-1} \tag{3.41}$$

So that the final expression for $\mathbf{W_{rx}}$ when using the Rx-ZF criterion becomes:

$$\mathbf{W_{rx}} = \left(\mathbf{W_{tx}}^H\mathbf{H}^H\mathbf{R_n}^{-1}\mathbf{H}\mathbf{W_{tx}}\right)^{-1}\mathbf{W_{tx}}^H\mathbf{H}^H\mathbf{R_n}^{-1} \tag{3.42}$$

3.3.3 Generalized Tx-ZF

As we said at the end of section 3.3.1, the idea for creating this new precoding scheme is to allow the equivalent channel matrix $\widetilde{\mathbf{H}}$ to be an arbitrary diagonal matrix, instead of the identity matrix. This actually means to allow each particual data stream in the system to experience a different attenuation. Mathematically, we replace condition (3.21) by:

$$\mathbf{y}_i^k = b_i^k\mathbf{u}_i^k + \mathbf{z}_i^k \tag{3.43}$$

Where, as we introduced in section 2.1, the index i refers to the user and k to the antenna or data stream.

Those extra degrees of freedom allow us to carry out a generalized optimization problem which will usually outperform the basic Tx-ZF scheme in terms of SNIR. In the worst case scenario, if the optimal choice for the set of b_i^k happens to be of the form $b_i^k = \alpha \; \forall \, i, k$ then the performance of Generalized Tx-ZF would match that of basic Tx-ZF but it will never be lower, as it is a generalization. To sum up, with respect to the simpler Tx-ZF, this scheme increases the total SNIR at the price of having unequal SNIR per data-stream. In a multi-user scenario, this implies that the distinct users will experience different Quality of Service (QoS), which may be or not a problem depending on the mobile operator's policy.

We must also note that, just like the Tx-ZF, this filter is thought with either a single-user or a downlink multi-user scenario in mind.

To obtain the generalized Tx-ZF precoder, we assume a known and constant $\mathbf{W_{rx}}$, just as for the Tx-ZF. However, since now $\mathbf{W_{rx}HW_{tx}} = \mathbf{B}$ with $\mathbf{B}$ an arbitrary diagonal matrix of size $Nr \times Nr$ obtained as $\mathbf{B} = \text{diag}(\mathbf{b}), \mathbf{b} = \left[b_1^1, \ldots, b_1^r, \ldots, b_N^1, \ldots, b_N^r\right]^T$, we will modify the receive filter to be $\mathbf{B}^{-1}\mathbf{W_{rx}}$, as shown in figure 3.1. The introduction of $\mathbf{B}^{-1}$ in the receiver simply serves the purpose of keeping the energy per symbol of the received constellation in the same order of magnitude as the transmitted constellation. This way, a direct comparison (for instance in terms of MSE) between $\hat{\mathbf{u}}$ and $\mathbf{u}$ is fair. Besides, the introduction of $\mathbf{B}^{-1}$ in the receiver is equivalent to forcing again $\widetilde{\mathbf{H}} = \mathbf{I}_{Nr}$, so that in the end the resulting MSE will be only due to the noise.

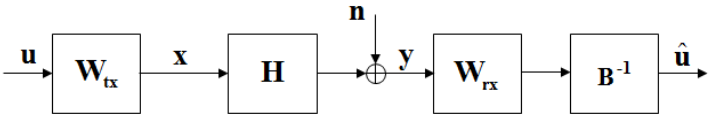


Figure 3.1: Generalized Tx-ZF system model

As we said, the introduction of $\mathbf{B}$ creates Nr new degrees of freedom which can be exploited to improve the design of the precoder with respect to the Tx-ZF. Without any loss of generality, we will assume that $b_i^k \in \mathbb{R}$.

We will formulate a optimization problem which tries to minimize the MSE under the constraint $\mathbf{W_{rx}HW_{tx}} = \mathbf{B}$, taking into account that there is a new element in the reception chain after $\mathbf{W_{rx}}$: the diagonal matrix $\mathbf{B}^{-1}$, which affects the MSE since it will modify the noise power. We also include the TPC directly into the optimization problem, unlike they do in [9] when deriving the Tx-ZF.

Mathematically, the optimization problem can be expressed as:

$$\min_{\mathbf{W_{tx}},\mathbf{b}} \mathbb{E}\left\{\left\|\mathbf{B}^{-1}\mathbf{W_{rx}n}\right\|^2\right\} \quad s.t. \quad \mathbf{W_{rx}HW_{tx}} = \mathbf{B}$$
$$\mathbb{E}\left\{\left\|\mathbf{W_{tx}u}\right\|^2\right\} \leq P_{\max} \tag{3.44}$$

Where we have used that, since there is no interference, the MSE amounts to simply the

total noise power at the output of the whole receive chain. The Lagrangian associated to that problem is:

$$\begin{aligned}\mathcal{L}(\mathbf{W_{tx}}, \mathbf{b}, \lambda, \mathbf{\Phi}) = \mathrm{Tr}\left(\mathbf{B}^{-1}\mathbf{W_{rx}}\mathbf{R_n}\mathbf{W_{rx}}^H\mathbf{B}^{-1}\right) + \lambda\left(\mathrm{Tr}\left(\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W_{tx}}^H\right) - P_{\max}\right) - \\ - \mathrm{Tr}\left(\left(\mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}} - \mathbf{B}\right)\mathbf{\Phi}^H\right)\end{aligned} \tag{3.45}$$

First of all, we will differentiate the Lagrangian with respect to $\mathbf{W_{tx}}$ to obtain:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W_{tx}}} = \lambda \mathbf{R_u}\mathbf{W_{tx}}^H - \mathbf{\Phi}^H\mathbf{W_{rx}}\mathbf{H} \tag{3.46}$$

Equating the derivative to zero and solving for $\mathbf{W_{tx}}$ we get:

$$\mathbf{W_{tx}} = \frac{1}{\lambda}\mathbf{H}^H\mathbf{W_{rx}}^H\mathbf{\Phi}\mathbf{R_u}^{-1} \tag{3.47}$$

Inserting the previous expression into the constraint $\mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}} = \mathbf{B}$ we have:

$$\begin{aligned}\mathbf{B} &= \frac{1}{\lambda}\mathbf{W_{rx}}\mathbf{H}\mathbf{H}^H\mathbf{W_{rx}}^H\mathbf{\Phi}\mathbf{R_u}^{-1} \\ \mathbf{\Phi} &= \lambda\left(\mathbf{W_{rx}}\mathbf{H}\mathbf{H}^H\mathbf{W_{rx}}^H\right)^{-1}\mathbf{B}\mathbf{R_u} \\ \mathbf{W_{tx}} &= \mathbf{H}^H\mathbf{W_{rx}}^H\left(\mathbf{W_{rx}}\mathbf{H}\mathbf{H}^H\mathbf{W_{rx}}^H\right)^{-1}\mathbf{B}\end{aligned} \tag{3.48}$$

At this point, we must differentiate the Lagrangian with respect each of the b_i^k too. In order to do that, we observe that the Lagrangian can be written as:

$$\mathcal{L}(\mathbf{W_{tx}}, \mathbf{b}, \lambda, \mathbf{\Phi}) = \sum_{i=1}^{N}\sum_{k=1}^{r}\left(\frac{\sigma_{\mathbf{z}_i^k}}{b_i^k}\right)^2 + \sum_{i=1}^{N}\sum_{k=1}^{r}\boldsymbol{\phi}_i^k b_i^k + f\left(\mathbf{W_{tx}}, \lambda\right) + g\left(\mathbf{W_{tx}}, \mathbf{\Phi}\right) \tag{3.49}$$

In the previous equation, we have introduced $\boldsymbol{\phi} \in \mathbb{R}^{Nr}$ as a vector containing the elements in the diagonal of $\mathbf{\Phi}$, that is, $\boldsymbol{\phi} = \mathrm{diag}(\mathbf{\Phi})$. Note that we are indexing the elements of $\boldsymbol{\phi}$ under an implicit partition $\boldsymbol{\phi} = [\boldsymbol{\phi}_1\boldsymbol{\phi}_2\ldots\boldsymbol{\phi}_N]^T, \boldsymbol{\phi}_i \in \mathbb{R}^r$, using the notation we defined in chapter 2. Besides, we have also introduced $\sigma_{\mathbf{z}_i^k}$ as the standard deviation of the equivalent noise $\mathbf{z}_i^k$. Recall that, since $\mathbf{z} = \mathbf{W_{rx}}\mathbf{n}$, the autocorrelation matrix of $\mathbf{z}$ is $\mathbf{R_z} = \mathbf{W_{rx}}\mathbf{R_n}\mathbf{W_{rx}}^H$. Then, $\sigma^2_{\mathbf{z}_i^k}$ can be found from the (Nr)-dimensional vector formed with the elements of the diagonal of $\mathbf{R_z}$ using the same indexing we employed for $\boldsymbol{\phi}$.

Assuming that $\mathbf{R_u}$ is a diagonal matrix, i.e. that the data streams are uncorrelated, we can find:

$$\boldsymbol{\phi}_i^k = \lambda\sigma^2_{\mathbf{u}_i^k}b_i^k\mathbf{c}_i^k \tag{3.50}$$

Where we have introduced the power of the symbol to be sent on the k-th data stream of the i-th user, $\sigma^2_{\mathbf{u}_i^k}$ indexed in the same way as $\sigma^2_{\mathbf{z}_i^k}$ but using the elements in the diagonal of

the autocorrelation matrix $\mathbf{R_u}$. More importantly, we have also defined the vector $\mathbf{c} \in \mathbb{R}^{Nr}$ as $\mathbf{c} = \text{diag}\left(\left(\mathbf{W_{rx}}\mathbf{H}\mathbf{H}^H\mathbf{W}_{\mathbf{rx}}^H\right)^{-1}\right)$ which is being indexed under the same scheme as $\boldsymbol{\phi}$.

If we substitute the value of $\boldsymbol{\phi}_i^k$ in the expression of the Lagrangian we obtained before:

$$\mathcal{L}(\mathbf{W_{tx}}, \mathbf{b}, \lambda, \mathbf{\Phi}) = \sum_{i=1}^{N}\sum_{k=1}^{r}\left(\frac{\sigma_{\mathbf{z}_i^k}}{b_i^k}\right)^2 + \lambda\sum_{i=1}^{N}\sum_{k=1}^{r}\sigma_{\mathbf{u}_i^k}^2(b_i^k)^2\mathbf{c}_i^k + f\left(\mathbf{W_{tx}}, \lambda\right) + g\left(\mathbf{W_{tx}}, \mathbf{\Phi}\right) \tag{3.51}$$

Taking the derivative with respect to b_i^k and equating it to zero we have:

$$\frac{\partial\mathcal{L}}{\partial b_i} = -2\frac{\sigma_{\mathbf{z}_i^k}^2}{(b_i^k)^3} + 2\lambda\sigma_{\mathbf{u}_i^k}^2 b_i^k\mathbf{c}_i^k = 0 \tag{3.52}$$

In the previous expression, we can solve for the value of b_i^k as a function of the dual variable λ:

$$\begin{aligned} b_i &= \frac{1}{\sqrt[4]{\lambda}}\tilde{b}_i \\ \tilde{b}_i &= \sqrt[4]{\frac{\sigma_{\mathbf{z}_i^k}^2}{\sigma_{\mathbf{u}_i^k}^2}\frac{1}{\mathbf{c}_i^k}} \\ \widetilde{\mathbf{B}} &= \text{diag}\left(\left[\tilde{b}_1^1, \tilde{b}_1^2, \ldots, \tilde{b}_1^r, \ldots, \tilde{b}_N^1, \tilde{b}_N^2, \ldots, \tilde{b}_N^r\right]\right) \end{aligned} \tag{3.53}$$

Finally, to obtain the value of λ we use, as always, the TPC by replacing $\mathbf{W_{tx}}$ and $\mathbf{B}$ with the expressions obtained previously. Doing that, we get:

$$\frac{1}{\sqrt[4]{\lambda}} = \sqrt{\frac{P_{\max}}{\text{Tr}\left(\widetilde{\mathbf{B}}\mathbf{R_u}\widetilde{\mathbf{B}}\left(\mathbf{W_{rx}}\mathbf{H}\mathbf{H}^H\mathbf{W}_{\mathbf{rx}}^H\right)^{-1}\right)}} \tag{3.54}$$

Then, the final expression for the generalized Tx-ZF precoding matrix is:

$$\mathbf{W_{tx}} = \sqrt{\frac{P_{\max}}{\text{Tr}\left(\widetilde{\mathbf{B}}\mathbf{R_u}\widetilde{\mathbf{B}}\left(\mathbf{W_{rx}}\mathbf{H}\mathbf{H}^H\mathbf{W}_{\mathbf{rx}}^H\right)^{-1}\right)}}\mathbf{H}^H\mathbf{W}_{\mathbf{rx}}^H\left(\mathbf{W_{rx}}\mathbf{H}\mathbf{H}^H\mathbf{W}_{\mathbf{rx}}^H\right)^{-1}\widetilde{\mathbf{B}} \tag{3.55}$$

With $\widetilde{\mathbf{B}}$ as it was defined in equation (3.53).

It is important to observe that the modification shown here with respect to the Tx-ZF scheme in [9] reduces to the original Tx-ZF in the particular case when the quotient $\frac{\sigma_{\mathbf{z}_i^k}^2}{\sigma_{\mathbf{u}_i^k}^2}$ and the elements in the diagonal of the matrix $\left(\mathbf{W_{rx}}\mathbf{H}\mathbf{H}^H\mathbf{W}_{\mathbf{rx}}^H\right)^{-1}$ do not depend on the indexes i, k. This actually occurs whenever all the data streams face the same channel, which in turn depends on the distribution of the eigenvalues of the channel matrix $\mathbf{H}$. The worst conditioned $\mathbf{H}$ is, the better the generalized Tx-ZF will be with respect to the original Tx-ZF in [9].

3.3.4 Block diagonalization: Spencer's ZF

In section 3.3.1 we presented an elementary formulation for a zero forcing precoder. After that, in the previous section, we showed that we can improve our design if we relax some of the zero-forcing constraints to gain extra degrees of freedom, all while still keeping the interference at user level non-existent. Continuing with that idea, we will show now that even a more relaxed formulation can be used leading to even better results.

We just showed that, actually, to have an interference-free system, we don't need the equivalent channel matrix to be the identity matrix but it suffices to have a diagonal matrix. But, actually, we do not even need that. As we actually said in equation (3.19), having a block-diagonal equivalent channel matrix is enough to satisfy the objective of a zero-forcing design. Based on this observation, Q. H. Spencer formulated in [15] a very general zero-forcing scheme that will be studied in detail for the remaining of this section. Because of the way Spencer's ZF is obtained, this method is very commonly denoted as Block Diagonalization (BD) in the specialized literate.

Unlike the other MIMO filtering and/or precoding schemes that have been explained in this project up to know, Spencer's ZF is the first one which is actually implemented in real-world systems. Even if it is not strictly optimal, as we are about to see, it achieves a really nice trade-off between performance and complexity. Because of that, many engineers have decided to introduce this scheme in their communication systems, [16].

First of all, Spencer's ZF is unlike other formulations we have just seen because it does not start with an optimization problem. Also, it is the first scheme which has a joint tx-rx design, meaning that we will obtain simultaneously the values of $\mathbf{W_{tx}}$ and $\mathbf{W_{rx}}$. Finally, as we will see, this formulation has another very nice property: it decouples the power allocation optimization problem of the design of $\mathbf{W_{tx}}$, thus getting extra degrees of freedom in our design.

Spencer's ZF starts by targeting the root of the zero-forcing concept. What we really want, in the most general form, is to get an equivalent channel matrix which is block-diagonal.

Recall the notation we introduced in chapter 2 for partitioning the channel matrix $\mathbf{H}$ in sets of r rows as:

$$\mathbf{H} = \begin{pmatrix} (\mathbf{H})_1 \\ (\mathbf{H})_2 \\ \vdots \\ (\mathbf{H})_N \end{pmatrix} \tag{3.56}$$

Also, we partitioned the precoding matrix $\mathbf{W_{tx}}$ in sets of r columns as:

$$\mathbf{W_{tx}} = \left((\mathbf{W_{tx}})^1 (\mathbf{W_{tx}})^2 \dots (\mathbf{W_{tx}})^N \right) \tag{3.57}$$

As a recap, we remind the reader that under those definitions we have $(\mathbf{H})_i \in \mathbb{C}^{r \times Mt}$ and $(\mathbf{W_{tx}})^j \in \mathbb{C}^{Mt \times r}$.

What we specifically look for can be mathematically expressed as:

$$(\mathbf{H})_i(\mathbf{W_{tx}})^j = \mathbf{0}_{r\times r} \quad \text{if} \quad i \neq j \tag{3.58}$$

That condition can be readily satisfied if we choose the columns of $(\mathbf{W_{tx}})^j$ to be vectors within the null space of the matrix:

$$\overline{\mathbf{H}}_j = \begin{pmatrix} (\mathbf{H})_1 \\ (\mathbf{H})_2 \\ \vdots \\ (\mathbf{H})_{j-1} \\ (\mathbf{H})_{j+1} \\ \vdots \\ (\mathbf{H})_N \end{pmatrix} \tag{3.59}$$

In words, what we are doing is to ensure an interference-free system by choosing the beamforming vectors of the j-th user within the span of the null space of the channel matrices seen by all the other users. Note that $\overline{\mathbf{H}}_j \in \mathbb{C}^{(N-1)r\times Mt}$ and, under the assumption that $Mt \geq Nr$ and that $\mathbf{H}$ is non-singular, it follows that the rank of $\overline{\mathbf{H}}_j$ will be $(N-1)r$, implying that the dimension of its null space will be exactly $Mt - (N-1)r > 0$. Hence, our problem does have a solution indeed.

There are many ways of finding vectors satisfying that. Indeed, since the null space of a matrix is a linear subspace, there are infinitely many possible choices to solve this problem. However, one of the simplest and most computationally efficient implementations involves using the SVD of the matrix $\overline{\mathbf{H}}_j$. As we know, any matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ can be expressed as:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H \tag{3.60}$$

Where $\mathbf{U} \in \mathbb{C}^{m\times m}$ and $\mathbf{V} \in \mathbb{C}^{n\times n}$ are unitary matrices and $\mathbf{\Sigma} \in \mathbb{C}^{m\times n}$ is a diagonal matrix whose diagonal entries are the singular values of the matrix $\mathbf{A}$ in decreasing order. If the rank of $\mathbf{A}$ is $\text{rank}(\mathbf{A}) = s$, then only the first s singular values will be non-zero. Moreover, there are several key properties that follow from the SVD. Defining the following partition of the matrices involved:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H = \Big((\mathbf{U})_{m\times s}(\mathbf{U})_{m\times m-s}\Big) \begin{pmatrix} (\mathbf{\Sigma})_{s\times s} & (\mathbf{0})_{s\times m-s} \\ (\mathbf{0})_{m-s\times s} & (\mathbf{0})_{m-s\times m-s} \end{pmatrix} \Big((\mathbf{V})_{n\times s}(\mathbf{V})_{n\times n-s}\Big)^H \tag{3.61}$$

The columns of $\mathbf{U}$ are usually referred to as left singular vectors, whereas columns of $\mathbf{V}$ are denoted right singular vectors. Upon the previous partition, we can realize that:

$$\mathbf{A} = (\mathbf{U})_{m\times s}(\mathbf{\Sigma})_{s\times s}(\mathbf{V})^H_{n\times s} \tag{3.62}$$

A form which is denoted reduced SVD.

In table 3.1 we summarize a set of well-known facts about the SVD will be useful along this section:

Columns of	**are an orthonormal basis of**
$(\mathbf{U})_{m\times s}$	Column space of $\mathbf{A}$
$(\mathbf{V})_{n\times s}$	Column space of $\mathbf{A}^H$
$(\mathbf{U})_{m\times m-s}$	Null space of $\mathbf{A}^H$
$(\mathbf{V})_{n\times n-s}$	Null space of $\mathbf{A}$

Table 3.1: Meaning behind SVD of an arbitrary $m \times n$ matrix $\mathbf{A}$ of rank s.

Those statements follow from elementary matrix algebra, hence the reader can find proof of any of them in any undergraduate text on algebra such as [17].

Going back to our work, if we compute the SVD of the matrix $\overline{\mathbf{H}}_j$:

$$\overline{\mathbf{H}}_j = \widetilde{\mathbf{U}}_j \tilde{\mathbf{\Sigma}}_j \widetilde{\mathbf{V}}_j^H \tag{3.63}$$

With $\widetilde{\mathbf{U}}_j \in \mathbb{C}^{(N-1)r\times(N-1)r}$, $\tilde{\mathbf{\Sigma}}_j \in \mathbb{C}^{(N-1)r\times Mt}$ and $\widetilde{\mathbf{V}}_j^H \in \mathbb{C}^{Mt\times Mt}$, then, we are interested in obtaining the last $Mt-(N-1)r$ right singular vectors, that is, the last $Mt-(N-1)r$ columns of $\widetilde{\mathbf{V}}_j^H$. This is because, denoting by $\mathbf{W}_j^1$ the $Mt \times Mt-(N-1)r$ matrix with those right singular vectors, we have that $\mathbf{W}_j^1$ then satisfies:

$$(\mathbf{H})_i \mathbf{W}_j^1 = \mathbf{0}_{r\times Mt-(N-1)r} \quad \text{if} \quad i \neq j \tag{3.64}$$

Which is exactly what we wanted. However, we are not fully done yet. In the end, we do not only want an scheme that guarantees that the equivalent channel matrix is block diagonal but, also, that each receiver can uncouple the different data streams directed to that user. In other words, we need to provide a way for the receiver to "undo" the coupling introducing by the matrices $(\mathbf{H})_j \mathbf{W}_j^1$, eliminating also the interference at antenna level and not only at user level. In order to do so, we will use the SVD yet once more. Let us consider now the reduced SVD of $(\mathbf{H})_j \mathbf{W}_j^1$:

$$(\mathbf{H})_j \mathbf{W}_j^1 = \mathbf{U}_j \mathbf{\Sigma}_j {\mathbf{V}^H}_j \tag{3.65}$$

Since $(\mathbf{H})_j \mathbf{W}_j^1 \in \mathbb{C}^{r\times Mt-(N-1)r}$ and we assume $(\mathbf{H})_j$ to have all its rows linearly independent, we see that the rank of $(\mathbf{H})_j \mathbf{W}_j^1$ is r. Also, as we are using now a reduced SVD form, we have that $\mathbf{U}_j \in \mathbb{C}^{r\times r}$, $(\mathbf{\Sigma})_j \in \mathbb{C}^{r\times r}$ and ${\mathbf{V}^H}_j \in \mathbb{C}^{r\times Mt-(N-1)r}$. Let us denote now $\mathbf{W}_j^2 = \mathbf{V}_j$. We must point out that the previous matrix is not square, hence, it is not unitary unlike in the non-reduced SVD. However, since the columns of that matrix are an orthonormal basis, in this case of the column space of $\left((\mathbf{H})_j \mathbf{W}_j^1\right)^H$, we will have that:

$$(\mathbf{H})_j \mathbf{W}_j^1 \mathbf{W}_j^2 = \mathbf{U}_j \mathbf{\Sigma}_j \mathbf{V}^H{}_j \mathbf{V}_j = \mathbf{U}_j \mathbf{\Sigma}_j \tag{3.66}$$

Similarly, $\mathbf{U}^H{}_j \mathbf{U}_j = \mathbf{I}$. At this point, the reader should notice where we are heading. We are going to use as precoding matrix:

$$(\mathbf{W_{tx}})^j = \mathbf{W}_j^1 \mathbf{W}_j^2 \tag{3.67}$$

With $\mathbf{W}_j^1$ being the last $Mt - (N-1)r$ right singular vectors of $\overline{\mathbf{H}}_j$ and $\mathbf{W}_j^2$ the first r right singular vectors of $(\mathbf{H})_j \mathbf{W}_j^1$. In this way, the resulting $(\mathbf{W_{tx}})^j$ has a size $Mt \times r$ as expected. In the same way, we will employ as receive filter:

$$(\mathbf{W_{rx}})_{j,j} = \mathbf{U}^H{}_j \tag{3.68}$$

Thanks to those decisions, we end up with the following end-to-end MIMO system:

$$\begin{aligned}
&\hat{\mathbf{u}} = \mathbf{W_{rx}} \mathbf{H} \mathbf{W_{tx}} \mathbf{u} + \mathbf{W_{rx}} \mathbf{n} = \\
&= \begin{pmatrix} \mathbf{U}^H{}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{U}^H{}_N \end{pmatrix} \begin{pmatrix} (\mathbf{H})_1 \\ \vdots \\ (\mathbf{H})_N \end{pmatrix} \left((\mathbf{W_{tx}})^1 \dots (\mathbf{W_{tx}})^N \right) \mathbf{u} + \mathbf{z} = \\
&= \begin{pmatrix} \mathbf{U}^H{}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{U}^H{}_N \end{pmatrix} \begin{pmatrix} \mathbf{U}_1 \mathbf{\Sigma}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{U}_N \mathbf{\Sigma}_N \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_N \end{pmatrix} + \begin{pmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_N \end{pmatrix} = \\
&= \begin{pmatrix} \mathbf{\Sigma}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{\Sigma}_N \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_N \end{pmatrix} + \begin{pmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_N \end{pmatrix}
\end{aligned} \tag{3.69}$$

Therefore, we have a set of N non-interacting channels of the form:

$$\hat{\mathbf{u}}_i = \mathbf{\Sigma}_i \mathbf{u}_i + \mathbf{z}_i \quad \Leftrightarrow \quad \hat{\mathbf{u}}_i^k = \sqrt{\lambda_i^k} \mathbf{u}_i^k + \mathbf{z}_i^k \tag{3.70}$$

Where the elements in the diagonal of $\mathbf{\Sigma}_i$ have been defined as:

$$\mathbf{\Sigma}_i = \begin{pmatrix} \sqrt{\lambda_i^1} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \sqrt{\lambda_i^r} \end{pmatrix} \tag{3.71}$$

Also, since $\mathbf{z}_i = \mathbf{U}^H{}_i \mathbf{n}_i$, the autocorrelation matrix of $\mathbf{z}_i$ becomes $\mathbf{R}_{\mathbf{z}_i} = \mathbf{U}^H{}_i \mathbf{R}_\mathbf{n} \mathbf{U}_i$. Besides, in the usual case when $\mathbf{R}_\mathbf{n} = \sigma_n^2 \mathbf{I}_{Nr}$, we can see that $\mathbf{R}_{\mathbf{z}_i}$ simplifies to $\mathbf{R}_{\mathbf{z}_i} = \sigma_n^2 \mathbf{I}_r$. On the other hand, for an arbitrary $\mathbf{R}_\mathbf{n}$, the relative noise power in each antenna may change, but the total noise power for the i-th user does not as $\mathrm{Tr}\left(\mathbf{R}_{\mathbf{z}_i}\right) = \mathrm{Tr}\left(\mathbf{R}_{\mathbf{n}_i}\right)$.

Since we have a fully diagonalized system model now, equation (2.47) breaks down to a much simpler form when we assume that $\mathbf{R_u}$ and $\mathbf{R_z}$ are diagonal matrices too:

$$R_i = \mathbb{E}\left\{\sum_{k=1}^{r} \log_2\left(1 + \frac{\lambda_i^k}{\sigma^2_{\mathbf{z}_i^k}}\sigma^2_{\mathbf{u}_i^k}\right)\right\} \tag{3.72}$$

Where we have used that the determinant of a diagonal matrix is simply the product of the diagonal entries and that the logarithm of a product is the sum of the logarithms. In addition to that, we employed again $\sigma^2_{\mathbf{u}_i^k}$ and $\sigma^2_{\mathbf{z}_i^k}$ to refer to the elements in the diagonal of $\mathbf{R}_{\mathbf{u}_i}$ and $\mathbf{R}_{\mathbf{z}_i}$ respectively.

It is also important to recall that the diagonal entries of $\mathbf{\Sigma}_i$, $\sqrt{\lambda_i^k}$, depend on the particular realization of the matrix $\mathbf{H}$. Therefore, in order to compute the total rate of the system, we must average expression (3.72) taking into account the probability density function of the channel matrix.

Due to the fact that we have been able to compute $\mathbf{W_{tx}}$ without worrying about the power allocation yet, we can use the already known value obtained for $\mathbf{W_{tx}}$ to formulate a TPC, a PBPC or even a PAPC in a very simple way. Actually, looking at equation (3.72), we see the effect we talked about in the introduction: we have uncoupled the power allocation problem from the design of $\mathbf{W_{tx}}$. Now all we have to do is to compute $\left\{\sigma^2_{\mathbf{u}_i^k}\right\}_{i=1,\dots,N\, k=1,\dots,r}$ such that whatever transmit power constraint we are considering is satisfied. In general, there will be several feasible choices for the power allocation. Hence, we may end up designing a new mathematical optimization problem to find the "best" feasible choice, according to some objective function to be optimized.

Its compatibility with any kind of transmit power constraints and the fact that coordination between receivers is not required, make BD a perfectly feasible scheme to be applied in distributed multi-user MIMO scenarios on a per-cluster basis. Besides, BD not only achieves our objective of having an interference-free system but in [15] they prove that, subject to a generalized zero-forcing constraint, this scheme maximizes the system's rate. Therefore, it should no longer surprising that BD is probably the most popular precoding scheme nowadays for cellular coordinated MIMO.

In the remaining of this section, we will study several power allocation methods which can be employed along with Spencer's ZF. We will consider a PBPC in all cases. This is because, as we discussed in section 2.1.2, a PAPC can be studied as a particular case of a PBPC. Similarly, a TPC can be studied as a PBPC when we have only one transmitter. Therefore, if we solve the problem with a PBPC, we have solved all cases at once.

Uniform power allocation

The naivest approach to the power allocation problem is to use uniform power allocation. This means that all the data streams of all transmitters are assigned exactly the same power. Mathematically:

$$\sigma^2_{\mathbf{u}_i^k} = \sigma_u^2 \quad \forall\, i = 1, \ldots, N \text{ and } k = 1, \ldots, r \tag{3.73}$$

In that case, the value of σ_u^2 needs to be chosen so that the total power transmitted by each of the base-stations, $P_{\text{tx,j}}$, does not surpass the maximum power available for that BTS, $P_{\text{max,j}}$. Doing so analytically is not possible. However, we can use a really simple numerical procedure which we are about to describe.

First of all, recall that we can compute the transmitted power per antenna in the system as:

$$\mathbf{p} = \mathbf{W_{tx}^2 p_u} \tag{3.74}$$

In the previous equation, $\mathbf{W_{tx}^2}$ and $\mathbf{p_u}$ are as defined in section 2.1.2. The vector $\mathbf{p} \in \mathbb{R}^{Mt}$ is such that $\mathbf{p}_j^l$ contains the power transmitted by the l-th antenna of the j-th base-station. Particularizing for a uniform power distribution, we have $\mathbf{p_u} = \sigma_u^2 \mathbf{1}_{Nr}$ with $\mathbf{1}_{Nr}$ being a constant vector with all its Nr entries with value 1. In order to compute the total transmitted power per BTS we recover the auxiliary matrix $\mathbf{A}$ defined in section 2.1.2 as $\mathbf{A} = \mathbf{I}_{M\times M} \otimes \mathbf{1}_t^T$ so that the power consumed by the j-th BTS is simply given by the j-th element of $\mathbf{Ap}$.

Then, we will try to find the biggest value of σ_u^2 such that the power transmitted by each BTSs is below their own power constraint. Mathematically, we look for:

$$\sigma_u^2 = \sup\left\{ x \mid x\left(\mathbf{A W_{tx}^2 1}_{Nr}\right)_j \leq P_{\text{max,j}} \quad \forall\, j = 1, \ldots, M \right\} \tag{3.75}$$

We can employ a bisection search to find the supremum we are looking for in a very short time. Even if clearly suboptimal, uniform power allocation is widely used by a very important reason: it is extremely simple to implement. Moreover, its performance is acceptable in most scenarios, as generally the cost incurred by including a mathematical optimization to solve the power allocation problem exceeds the benefits produced by the slight improvement in the system's performance we would obtain.

Optimal power allocation

If employing a uniform power allocation is the simplest solution we can use, the opposite is to use another optimization problem, this time with the power allocation vector $\mathbf{p_u}$ as the variable to be optimized. The most sensible optimization problem we can propose is a maximization of the total rate of the system, that is:

$$\max_{\mathbf{p_u}} R(\mathbf{p_u}) = \sum_{i=1}^{N}\sum_{k=1}^{r} \log_2\left(1 + \frac{\lambda_i^k}{\sigma^2_{\mathbf{z}_i^k}} \sigma^2_{\mathbf{u}_i^k}\right) \quad s.t. \quad \mathbb{E}\left\{ \left\| (\mathbf{W_{tx}})_j \mathbf{u} \right\|^2 \right\} \leq P_{\text{max,j}} \tag{3.76}$$

Where we have used a partition in sets of t rows of the precoding matrix $\mathbf{W_{tx}}$, as defined in section 2.1. As a consequence, recall that $(\mathbf{W_{tx}})_j \in \mathbb{C}^{t\times Nr}\ \forall\, j = 1, \ldots, M$.

Since the logarithm of an affine function of x is concave on x, and a sum of concave functions is concave, we can conclude that our current objective function, the sum rate in the system R, is a concave function of $\mathbf{p_u}$. Moreover, the power constraints $\mathbb{E}\left\{\|(\mathbf{W_{tx}})_j\mathbf{u}\|^2\right\} \leq P_{\text{max,j}}$ can be expressed as $\left(\mathbf{AW_{tx}^2 p_u}\right)_j \leq P_{\text{max,j}}\ \forall j = 1,\ldots,M$ as we saw in section 2.1.2 and during the explanation of the uniform power allocation algorithm in the previous subsection. Therefore, the constraints are linear on $\sigma^2_{\mathbf{u}_i^k}$.

As a consequence, the previous optimization problem is convex and its solution can be found employing numerical methods based on convex optimization. Several suitable libraries for MATLAB can be used for that purpose, like cvx. However, in order to get a higher grade of insight about the problem, we will try to find an analytical solution to that problem using the KKT conditions.

The Lagrangian can be written as:

$$\mathcal{L}(\mathbf{p_u}, \boldsymbol{\mu}) = R(\mathbf{p_u}) - \boldsymbol{\mu}^T\left(\mathbf{AW_{tx}^2 p_u} - \mathbf{P_{max}}\right) \tag{3.77}$$

Where we have introduced the vector of Lagrangian multipliers:

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_M \end{pmatrix} \tag{3.78}$$

And the vector of maximum powers per BTS:

$$\mathbf{P_{max}} = \begin{pmatrix} P_{\text{max,1}} \\ \vdots \\ P_{\text{max,M}} \end{pmatrix} \tag{3.79}$$

The derivative of the Lagrangian with respect to $\sigma^2_{\mathbf{u}_i^k}$ can be written as:

$$\frac{\partial\mathcal{L}(\mathbf{p_u}, \boldsymbol{\mu})}{\partial\sigma^2_{\mathbf{u}_i^k}} = \frac{\frac{\lambda_i^k}{\sigma^2_{\mathbf{z}_i^k}}}{\ln(2)\left(\frac{\lambda_i^k}{\sigma^2_{\mathbf{z}_i^k}}\right)\sigma^2_{\mathbf{u}_i^k}} - \mathbf{l}_i^k \tag{3.80}$$

Where we have introduced $\mathbf{l} \in \mathbb{R}^{Nr}$ defined as $\mathbf{l} = \left(\boldsymbol{\mu}^T\mathbf{AW_{tx}^2}\right)^T$ and we have employed the same vector indexing scheme we have been extensively using throughout this chapter. Note that $\boldsymbol{\mu}^T\mathbf{AW_{tx}^2}$ is a row vector of size Nr so that the $k+(i-1)r$-th entry, which under our indexing scheme is denoted $\mathbf{l}_i^k$, accounts for the slope of the affine function $\boldsymbol{\mu}^T\left(\mathbf{AW_{tx}^2 p_u} - \mathbf{P_{max}}\right)$ with respect to independent the variable $\sigma^2_{\mathbf{u}_i^k}$.

Equating the previous derivative to zero, this can be rearranged into the following form:

$$\sigma^2_{\mathbf{u}_i^k} + \left(\frac{\lambda_i^k}{\sigma^2_{\mathbf{z}_i^k}}\right)^{-1} = \frac{1}{\ln(2)\mathbf{l}_i^k} \tag{3.81}$$

The reader can now spot the great similarity between the solution obtained in equation (3.81) and the well-known waterfilling distribution. The main difference is that, in this case, the waterlevel is given by $\frac{1}{\ln(2)\mathbf{l}_i^k}$. Therefore, the waterlevel is not fixed, but it depends on the particular data stream indexes i and k. Also, we see that the Lagrange multipliers μ_i appear coupled in the solution along with the waterlevels. We can write Nr equations of the form in (3.81) which, together with the M PBPC equations, suffice to find the unique optimum for all the $Nr + M$ variable (Nr power allocation coefficients $\sigma^2_{\mathbf{u}_i^k}$ and M Lagrange multipliers μ_i). However, due to the variability of the waterlevels and the coupling of the Lagrange multipliers, no closed-form solution exists and we must end up employing numerical optimization methods based on convex optimization to reach the particular solution. We could use convex optimization techniques to directly find the solution of the problem from the beginning, or we could use them to find the waterlevels and, then, obtain the optimal power allocation from (3.81). Either way, the complexity is similar.

Suboptimal power allocation schemes based on waterfilling

Even though the complexity of the optimal solution obtained before may be still too high for certain scenarios where including a convex-optimization routine in the software may be unfeasible, it serves as a motivation to develop suboptimal schemes. As we have seen, the optimal solution resembles a sort of variable-level waterfilling where the waterlevel was different for each particular data stream and the Lagrange multipliers were coupled in the set of equations.

On [18], the authors propose two different suboptimal power-allocation schemes based on waterfilling. Those allow obtaining a solution with traditional waterfilling algorithms which performs very closely to the optimal power allocation.

Modified waterfilling

The first algorithm proposed will try to reduce complexity by solving the coupling of the Lagrange multipliers in the equations shown before. In order to do that, the authors try to find a unique power constraint, more restrictive than all the original constraints, so that the resulting Lagrangian would have just one Lagrange multiplier μ.

In order to find that new "super-constraint", they propose introducing a virtual "equivalent" BTS which considers, for each data stream, the most stringent constraint of the original set. Mathematically, they define:

$$\Omega_j = \max_{k=1,\dots,M} \left(\mathbf{AW^2_{tx}}\right)^{k,j} \ \forall\, j = 1, \dots, Nr \tag{3.82}$$

From an intuitive point of view, what we are doing is as follows. The $M \times Nr$ matrix $\mathbf{AW^2_{tx}}$, when multiplied by the power allocation vector $\mathbf{p_u}$, gives a vector of size M with the power transmitted by each of the M BTSs. Instead of doing that, we collapse the M rows in $\mathbf{AW^2_{tx}}$ to a unique row by keeping, for each column, the entry with the biggest value out of the M entries in that column. In this way, we get a row vector $\mathbf{\Omega}$ of size Nr which is actually a worst-case combination of the M original rows. Since each of the original rows corresponded

to a particular BTS, what we are doing is constructing a virtual, worst-case BTS which suffers worse conditions from the power-allocation point of view than all of the M original BTSs.

With $\mathbf{\Omega}$ defined as above, the power constraint becomes:

$$\mathbf{\Omega}^T \mathbf{p_u} \leq P_{\max} \tag{3.83}$$

In this particular case, the authors assumed that all the BTSs had the same power available. If that was not the case, we could make:

$$P_{\max} = \min_{j=1,\ldots,M} P_{\max,\mathrm{j}} \tag{3.84}$$

In order to ensure that all the constraints are satisfied.

The readers can easily convince themselves that, as long as equation (3.83) holds, all of the original power constraints $\left(\mathbf{A}\mathbf{W}_{\mathbf{tx}}^{\mathbf{2}}\mathbf{p_u}\right)_j \leq P_{\max,\mathrm{j}}$ will hold too. Of course, the price we pay is that, except in very special realizations of the channel matrix, none of the original constraints will be active. In other words, this suboptimal approach is a bit too conservative, so that the total power employed will be smaller than the optimum.

Using the new "super-constraint", we can rewrite the Lagrangian as:

$$\mathcal{L}(\mathbf{p_u}, \mu) = R(\mathbf{p_u}) - \mu\left(\mathbf{\Omega}^T \mathbf{p_u} - P_{\max}\right) \tag{3.85}$$

The derivative with respect to $\sigma^2_{\mathbf{u}_i^k}$ is:

$$\frac{\partial \mathcal{L}(\mathbf{p_u}, \mu)}{\partial \sigma^2_{\mathbf{u}_i^k}} = \frac{\frac{\lambda_i^k}{\sigma^2_{\mathbf{z}_i^k}}}{\ln(2)\left(\frac{\lambda_i^k}{\sigma^2_{\mathbf{z}_i^k}}\right)\sigma^2_{\mathbf{u}_i^k}} - \mu\mathbf{\Omega}_i^k \tag{3.86}$$

Equating to 0 and rearranging as before:

$$\sigma^2_{\mathbf{u}_i^k} + \left(\frac{\lambda_i^k}{\sigma^2_{\mathbf{z}_i^k}}\right)^{-1} = \frac{1}{\ln(2)\mu\mathbf{\Omega}_i^k} \tag{3.87}$$

We can see that the problem is the same as finding the constant $K_{MWF} = \frac{1}{\ln(2)\mu}$ such that, for all the $\sigma^2_{\mathbf{u}_i^k}$, the following equation holds:

$$\sigma^2_{\mathbf{u}_i^k} = \left[K_{\mathrm{MWF}}\frac{1}{\mathbf{\Omega}_i^k} - \left(\frac{\lambda_i^k}{\sigma^2_{\mathbf{z}_i^k}}\right)^{-1}\right]^+ \tag{3.88}$$

Where the operator $[\cdot]^+$ is the identity function when its argument is positive and zero otherwise.

The reader can note that, again, this problem is a waterfilling distribution with a variable waterlevel in each of the datastreams. However, thanks to the clever trick of intruducing this "super-constraint", the authors in [18] have been able to successfully decouple the problem from

the Lagrange multipliers. In this new scenario, we only need to find the value of K_{MWF} and the $\sigma^2_{\mathbf{u}_i^k}$ since all of the $\mathbf{\Omega}_i^k$ are known. In order to solve that, algorithms similar to the ones which solve the standard waterfilling problem can be employed.

Even if suboptimal, numerical results confirm that this kind of power allocation can perform very close to the optimal, with a much lower computational complexity which avoids the burden of employing convex optimization routines every time the power allocation has to be refreshed.

Waterfilling

Also in [18], the authors propose yet another simplification. Taking into account that, in practical realizations, the distinct Ω_j are very similar for all j, we can approximate them as a constant which can be included directly into the waterlevel. Mathematically, this means:

$$\Omega_j \approx \Omega \; \forall \, j = 1, \ldots, Nr \tag{3.89}$$

$$K_{\text{WF}} = \frac{1}{\ln(2)\mu\Omega} \tag{3.90}$$

With this, we get a standard waterfilling problem:

$$\sigma^2_{\mathbf{u}_i^k} = \left[K_{\text{WF}} - \left(\frac{\lambda_i^k}{\sigma^2_{\mathbf{z}_i^k}} \right)^{-1} \right]^+ \tag{3.91}$$

For this scheme the complexity of finding the (suboptimal) solution is trivial, at the cost of a slightly greater performance gap with respect to the Modified Waterfilling scheme shown before.

3.4 MMSE filters

3.4.1 Rx-WF

The problem of finding the receive linear filter which minimizes the MSE is a well-known problem, which was solved by the American mathematician Norbert Wiener, [19]. In a completely general fashion, the solution to the problem of estimating a random vector $\mathbf{u}$ from another input random vector, $\mathbf{y}$, by using linear filtering as $\hat{\mathbf{u}} = \mathbf{W_{rx}}\mathbf{y}$ is given by:

$$\mathbf{W_{rx}} = \mathbf{R}_\mathbf{y}^{-1}\mathbf{R_{yu}} \tag{3.92}$$

Where $\mathbf{R_y}$ is the autocorrelation matrix of the input vector $\mathbf{y}$ and $\mathbf{R_{yu}}$ is the cross-correlation matrix between the input vector $\mathbf{y}$ and the vector to be estimated, $\mathbf{u}$. Applying (2.1) we have that:

$$\begin{aligned} \mathbf{R_y} &= \mathbf{H}\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W}_\mathbf{tx}^H\mathbf{H}^H + \mathbf{R_n} \\ \mathbf{R_{yu}} &= \mathbf{H}\mathbf{W_{tx}}\mathbf{R_u} \end{aligned} \tag{3.93}$$

So that the expression of the Rx-WF is simply:

$$\mathbf{W_{rx}} = (\mathbf{H}\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W_{tx}}^H\mathbf{H}^H + \mathbf{R_n})^{-1}\mathbf{H}\mathbf{W_{tx}}\mathbf{R_u} \tag{3.94}$$

Alternatively, using Woodbury's identity, also known as the matrix inversion lemma, we can write:

$$\mathbf{W_{rx}} = (\mathbf{W_{tx}}^H\mathbf{H}^H\mathbf{R_n}^{-1}\mathbf{H}\mathbf{W_{tx}} + \mathbf{R_u}^{-1})^{-1}\mathbf{W_{tx}}^H\mathbf{H}^H\mathbf{R_n}^{-1} \tag{3.95}$$

Both formulas for $\mathbf{W_{rx}}$ are fully equivalent and can be indistinctly used.

This linear filter as obtained here needs access to the signal in all the receive antennas, hence it requires full coordination in the receiver. The Rx-WF does not employ any kind of precoder. The matrix $\mathbf{W_{tx}}$ shown in the formulations is assumed fixed and known a priori when designing the receive filter and can perfectly be assumed to be an identity. Hence, as far as the Rx-WF is concerned, no coordination is needed in the transmitter. This scheme can be used then in single-user scenarios and in multi-user scenarios for the uplink channel only.

3.4.2 MMSE-UC

We will start studying MMSE precoders by what probably is the simplest possible approach: getting the value of $\mathbf{W_{tx}}$ which minimizes the total MSE without considering any power constraint at all and incorporating later the TPC with an heuristic approach like the one we used in section 3.3.1. This scheme will be denoted as Unconstrained MMSE precoder (MMSE-UC)

$$\min_{\mathbf{W_{tx}}} \mathbb{E}\left\{\|\mathbf{u} - \hat{\mathbf{u}}\|^2\right\} \tag{3.96}$$

In other words, we look for the value of $\mathbf{W_{tx}}$ which minimizes the trace of $\mathbf{R_e}$. Using (2.27) the cost function to be optimized can be written as:

$$f(\mathbf{W_{tx}}) = \mathrm{Tr}((\mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}} - \mathbf{I})\mathbf{R_u}\,(\mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}} - \mathbf{I})^H) + \mathrm{Tr}(\mathbf{W_{rx}}\mathbf{R_n}\mathbf{W_{rx}}^H) \tag{3.97}$$

Differentiating $f(\mathbf{W_{tx}})$ with respect to $\mathbf{W_{tx}}^H$ yields:

$$\frac{\partial f(\mathbf{W_{tx}})}{\partial \mathbf{W_{tx}}^H} = (\mathbf{W_{rx}}\mathbf{H})^H\,(\mathbf{W_{rx}}\mathbf{H})\,\mathbf{W_{tx}}\mathbf{R_u} - (\mathbf{W_{rx}}\mathbf{H})^H\,\mathbf{R_u} \tag{3.98}$$

Equating the derivative to 0 and solving for $\mathbf{W_{tx}}$ we get:

$$\mathbf{W_{tx}} = \Big((\mathbf{W_{rx}}\mathbf{H})^H\,(\mathbf{W_{rx}}\mathbf{H})\Big)^{-1}\,(\mathbf{W_{rx}}\mathbf{H})^H \tag{3.99}$$

If we look at equation (3.30), we can see that the solution for the Tx-ZF (before adding the heuristic scalar γ to satisfy the TPC) and the expression we have achieved now, also before satisfying the TPC are, indeed, very much alike. Not only they seem similar but, actually, they

are the same matrix. To prove that, let as consider an SVD decomposition of the equivalent channel matrix as seen by the precoder, $\mathbf{W_{rx}H} = \mathbf{U\Sigma V}^H$ and substitute into equations (3.30) and (3.99):

$$\begin{aligned}\left((\mathbf{W_{rx}H})^H(\mathbf{W_{rx}H})\right)^{-1}(\mathbf{W_{rx}H})^H &= \left(\mathbf{V\Sigma}^H\mathbf{U}^H\mathbf{U\Sigma V}^H\right)^{-1}\mathbf{V\Sigma}^H\mathbf{U}^H \\ &= \left(\mathbf{V}|\mathbf{\Sigma}|^2\mathbf{V}^H\right)^{-1}\mathbf{V\Sigma}^H\mathbf{U}^H \\ &= \mathbf{V\Sigma}^{-1}\mathbf{U}^H \end{aligned} \tag{3.100}$$

$$\begin{aligned}(\mathbf{W_{rx}H})^H\left((\mathbf{W_{rx}H})(\mathbf{W_{rx}H})^H\right)^{-1} &= \mathbf{V\Sigma}^H\mathbf{U}^H\left(\mathbf{U\Sigma V}^H\mathbf{V\Sigma}^H\mathbf{U}^H\right)^{-1} \\ &= \mathbf{V\Sigma}^H\mathbf{U}^H\left(\mathbf{U}|\mathbf{\Sigma}|^2\mathbf{U}^H\right)^{-1} \\ &= \mathbf{V\Sigma}^{-1}\mathbf{U}^H \end{aligned} \tag{3.101}$$

Therefore, both expressions actually define the same matrix.

As we can see, the point about this section was not studying a new filtering scheme, as we said at the beginning, but rather to shed some light about the characteristics of the Tx-ZF. Here we have learnt that such scheme is not just a basic Zero Forcing solution obtained by minimizing the total transmitted power while keeping the equivalent channel matrix as an identity matrix. In addition to that, it is the matrix which minimizes the total MSE when no transmit power constraints are directly included into the formulation.

3.4.3 MMSE-C

Taking a step forward with respect to the previous formulation, we can try to produce a similar MMSE formulation (fixed and known $\mathbf{W_{rx}}$, full coordination in the transmitter, no coordination in the receiver and TPC) but, this time, we will include the TPC directly into the optimization problem. This gives rise to our next MSSE scheme, the Constrained MMSE precoder (MMSE-C)

We can formulate the optimization problem as:

$$\min_{\mathbf{W_{tx}}} \mathbb{E}\left\{\|\mathbf{u}-\hat{\mathbf{u}}\|^2\right\} \quad s.t. \quad \mathbb{E}\left\{\|\mathbf{W_{tx}u}\|^2\right\} \leq P_{\max} \tag{3.102}$$

The Lagrangian for the previous problem is:

$$\begin{aligned}\mathcal{L}(\mathbf{W_{tx}},\lambda) = \mathrm{Tr}\left((\mathbf{W_{rx}HW_{tx}}-\mathbf{I})\,\mathbf{R_u}\,(\mathbf{W_{rx}HW_{tx}}-\mathbf{I})^H\right) + \\ + \mathrm{Tr}\left(\mathbf{W_{rx}R_nW}_{\mathbf{rx}}^H\right) + \lambda\left(\mathrm{Tr}(\mathbf{W_{tx}R_uW}_{\mathbf{tx}}^H - P_{\max}\right)\end{aligned} \tag{3.103}$$

Computing the derivative with respect to $\mathbf{W}_{\mathbf{tx}}^H$ yields:

$$\frac{\partial \mathcal{L}(\mathbf{W_{tx}}, \lambda)}{\partial \mathbf{W}_{\mathbf{tx}}^H} = (\mathbf{W_{rx}H})^H (\mathbf{W_{rx}H}) \mathbf{W_{tx}R_u} + \lambda \mathbf{W_{tx}R_u} - (\mathbf{W_{rx}H})^H \mathbf{R_u} \tag{3.104}$$

Equating the previous result to 0 and solving for $\mathbf{W_{tx}}$ we can get:

$$\mathbf{W_{tx}} = \left((\mathbf{W_{rx}H})^H (\mathbf{W_{rx}H}) + \lambda \mathbf{I} \right)^{-1} (\mathbf{W_{rx}H})^H \tag{3.105}$$

Using, as we did in the previous section, the SVD $\mathbf{W_{rx}H} = \mathbf{U\Sigma V}^H$, we can find that the previous expression for $\mathbf{W_{tx}}$ is:

$$\mathbf{W_{tx}} = \left(\mathbf{V}|\mathbf{\Sigma}|^2\mathbf{V}^H + \lambda \mathbf{I} \right)^{-1} \mathbf{V\Sigma}^H\mathbf{U}^H = \mathbf{V} \frac{\mathbf{\Sigma}^H}{|\mathbf{\Sigma}|^2 + \lambda \mathbf{I}} \mathbf{U}^H \tag{3.106}$$

Where we have introduced an abuse of notation which we will use through this project: the division of two diagonal matrices is to be understood as the diagonal matrix whose diagonal elements are obtained by the element-wise division of the diagonal entries of the matrix in the numerator and the diagonal entries of the matrix in the denominator.

If we do the following algebraic manipulation:

$$\begin{aligned} \mathbf{W_{tx}} &= \mathbf{V} \frac{\mathbf{\Sigma}^H}{|\mathbf{\Sigma}|^2 + \lambda \mathbf{I}} \mathbf{U}^H = \mathbf{V\Sigma}^H\mathbf{U}^H\mathbf{U}(\left(|\mathbf{\Sigma}|^2 + \lambda \mathbf{I} \right)^{-1} \mathbf{U}^H = \\ &= \mathbf{V\Sigma}^H\mathbf{U}^H(\left(\mathbf{U}|\mathbf{\Sigma}|^2\mathbf{U}^H + \lambda \mathbf{I} \right)^{-1} = \\ &= (\mathbf{W_{rx}H})^H \left((\mathbf{W_{rx}H}) (\mathbf{W_{rx}H})^H + \lambda \mathbf{I} \right)^{-1} \end{aligned} \tag{3.107}$$

In the end, we reach two equivalent expressions for $\mathbf{W_{tx}}$. However, due to the restriction $Nr \leq Mt$, the second form will involve the inversion of a smaller matrix $Nr \times Nr$ instead of $Mt \times Mt$, so it provides computational advantages.

We still need to figure out the value of the Lagrange multiplier λ. As usual, we can do so by using the TPC. However, unlike in previous formulations where we could easily solve for λ from the TPC, now there is no analytic formula which finds its value because the Lagrange multiplier is within an inverse of a sum of matrices. Therefore, in the most general case, we need to use numerical methods to find a positive root of the function:

$$f(\lambda) = \mathrm{Tr}\left(\mathbf{W_{tx}}(\lambda) \mathbf{R_u} \mathbf{W}_{\mathbf{tx}}^H(\lambda) \right) - P_{\max} \tag{3.108}$$

However, as we will show now, we can greatly simplify that expression if we assume, as it the most usual case, that $\mathbf{R_u} = \mathbf{I}$. The matrix $(\mathbf{W_{rx}H}) (\mathbf{W_{rx}H})^H$ is a hermitian matrix, hence it is a normal matrix and, according to the spectral theorem, it can be unitarily diagonalized as:

$$(\mathbf{W_{rx}H}) (\mathbf{W_{rx}H})^H = \mathbf{UDU}^H \tag{3.109}$$

Where $\mathbf{D}$ is related with the SVD of $\mathbf{W_{rx}H}$ as $\mathbf{D} = |\mathbf{\Sigma}|^2$. Furthermore, let us introduce for notational simplicity $\mathbf{F} = (\mathbf{W_{rx}H}) (\mathbf{W_{rx}H})^H + \lambda \mathbf{I}$. Using the previously defined SVD, we

can write that $\mathbf{F} = \mathbf{U}\left(\mathbf{D} + \lambda\mathbf{I}\right)\mathbf{U}^H$ and $\mathbf{F}^{-1} = \mathbf{U}\left(\mathbf{D} + \lambda\mathbf{I}\right)^{-1}\mathbf{U}^H$, expression which will come handy very soon.

On the other hand, inserting the form of $\mathbf{W_{tx}}$ obtained in (3.107), the term $\mathrm{Tr}\left(\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W_{tx}}^H\right)$ can be expressed as:

$$\begin{aligned}\mathrm{Tr}\left(\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W_{tx}}^H\right) &= \mathrm{Tr}\left(\left(\mathbf{W_{rx}}\mathbf{H}\right)^H\mathbf{F}^{-1}\mathbf{R_u}\mathbf{F}^{-1}\left(\mathbf{W_{rx}}\mathbf{H}\right)\right) = \\ &= \mathrm{Tr}\left(\mathbf{F}^{-1}\left(\mathbf{W_{rx}}\mathbf{H}\right)\left(\mathbf{W_{rx}}\mathbf{H}\right)^H\mathbf{F}^{-1}\mathbf{R_u}\right) = \\ &= \mathrm{Tr}\left(\mathbf{U}\frac{\mathbf{D}}{\left(\mathbf{D}+\lambda\mathbf{I}\right)}\mathbf{U}^H\mathbf{R_u}\right)\end{aligned} \tag{3.110}$$

Using the simplification $\mathbf{R_u} = \mathbf{I}$ we have that:

$$\mathrm{Tr}(\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W_{tx}}^H) = \sum_{k=1}^{Nr}\frac{\sigma_k^2(\mathbf{W_{rx}}\mathbf{H})}{\left(\sigma_k^2(\mathbf{W_{rx}}\mathbf{H})+\lambda\right)} \tag{3.111}$$

Where $\sigma_k^2(\mathbf{W_{rx}}\mathbf{H})$ is the k-th singular value squared of the matrix $\mathbf{W_{rx}}\mathbf{H}$ or, equivalently, the k-th eigenvalue of either the matrix $\left(\mathbf{W_{rx}}\mathbf{H}\right)\left(\mathbf{W_{rx}}\mathbf{H}\right)^H$ or the matrix $\left(\mathbf{W_{rx}}\mathbf{H}\right)^H\left(\mathbf{W_{rx}}\mathbf{H}\right)$.

As we see, now the problem of finding λ simplifies into finding the roots of a polynomial of order $2Nr$. Still, whenever $2Nr \geq 5$, the problem cannot be solved analytically. Nevertheless, in terms of computational complexity, this problem is much easier to solve than the more general form obtained before, which involves the computation of matrix inverses every time the objective function has to be evaluated.

When $\mathbf{W_{tx}}$ is computed according to the MMSE-C filter, we can reduce the expression of the end-to-end MIMO system to the following form:

$$\hat{\mathbf{u}} = \mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}}\mathbf{u} + \mathbf{W_{rx}}\mathbf{n} = \mathbf{U}\frac{\mathbf{D}}{\mathbf{D}+\lambda\mathbf{I}}\mathbf{U}^H\mathbf{u} + \mathbf{z} \tag{3.112}$$

The matrix $\mathbf{U}$ is unitary, therefore it can be interpreted as a rotation in the complex plane or a change of basis. Then, we see that the performance of the i-th subchannel depends on the value of the quotient $\frac{\sigma_k^2(\mathbf{W_{rx}}\mathbf{H})}{\sigma_k^2(\mathbf{W_{rx}}\mathbf{H})+\lambda}$.

We can also find a simplified expression of the error autocorrelation matrix, $\mathbf{R_e}$. Substituting the value of $\mathbf{W_{tx}}$ we can get:

$$\mathbf{R_e} = \left(\left(\mathbf{W_{rx}}\mathbf{H}\right)\left(\mathbf{W_{rx}}\mathbf{H}\right)^H + \lambda\mathbf{I}\right)^{-1}\mathbf{R_u}\left(\left(\mathbf{W_{rx}}\mathbf{H}\right)\left(\mathbf{W_{rx}}\mathbf{H}\right)^H + \lambda\mathbf{I}\right)^{-1} + \mathbf{W_{rx}}\mathbf{R_n}\mathbf{W_{rx}}^H \tag{3.113}$$

With this expression for $\mathbf{R_e}$, we can obtain performance measures such as the MSE and so on.

3.4.4 Tx-WF

In this section, we will study the derivation of a state-of-the-art MMSE precoder for MIMO systems. This precoder was proposed in [9]. The derivations shown during this section will be essential for our subsequent work since the most elaborated original contributions of this project on the design of MIMO precoders try to generalize and enhance the scheme we will be discussing now.

In [9], they formulate the problem of finding a MMSE precoder with a TPC, assuming full coordination in the transmitter and no coordination in the receiver. Therefore, a priori, they try to solve the same problem we have tackled in section 3.4.3. The difference lies in the introduction of a clever modification of the problem which will allow finding an analytic expression for the Lagrange multiplier.

For this, we will assume that after the receive filter $\mathbf{W_{rx}}$, which is once more assumed fixed and known by the transmitter, there is an Automatic Gain Control filter which amounts simply to an scalar α. In other words, what we are saying is that the end-to-end system is now characterized by the equation:

$$\hat{\mathbf{u}} = \alpha \mathbf{W_{rx}} \mathbf{H} \mathbf{W_{tx}} \mathbf{u} + \alpha \mathbf{W_{rx}} \mathbf{n} \tag{3.114}$$

A block-diagram representation of the new system model is depicted in figure 3.2.

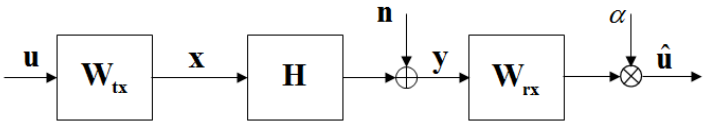


Figure 3.2: Tx-WF system model.

The optimization problem will be analogous to the one in section 3.4.3 but now, instead of optimizing only $\mathbf{W_{tx}}$, we will jointly optimize α and $\mathbf{W_{tx}}$. Mathematically:

$$\min_{\mathbf{W_{tx}},\alpha} \mathbb{E}\left\{\|\mathbf{u} - \hat{\mathbf{u}}\|^2\right\} \quad s.t. \quad \mathbb{E}\left\{\|\mathbf{W_{tx}}\mathbf{u}\|^2\right\} \leq P_{\max} \tag{3.115}$$

After the inclusion of α, the error autocorrelation matrix can be found by changing in (2.27) $\mathbf{W_{rx}}$ by $\alpha\mathbf{W_{rx}}$. Doing that we get:

$$\mathbf{R_e} = (\mathbf{I} - \alpha\mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}})\,\mathbf{R_u}\,(\mathbf{I} - \alpha\mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}})^H + |\alpha|^2 \mathbf{W_{rx}}\mathbf{R_n}\mathbf{W_{rx}^H} \tag{3.116}$$

Therefore, the Lagrangian associated to the constrained optimization problem takes the following form:

$$\begin{aligned}\mathcal{L}(\mathbf{W_{tx}}, \alpha, \lambda) = \operatorname{Tr}\left((\mathbf{I} - \alpha\mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}})\,\mathbf{R_u}\,(\mathbf{I} - \alpha\mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}})^H\right) + |\alpha|^2 \operatorname{Tr}\left(\mathbf{W_{rx}}\mathbf{R_n}\mathbf{W_{rx}^H}\right) + \\ + \lambda\left(\operatorname{Tr}\left(\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W_{tx}^H}\right) - P_{\max}\right)\end{aligned} \tag{3.117}$$

As usual, we will begin by differentiating the Lagrangian with respect to $\mathbf{W}_{\mathbf{tx}}^H$ to find:

$$\frac{\partial \mathcal{L}(\mathbf{W_{tx}}, \alpha, \lambda)}{\partial \mathbf{W}_{\mathbf{tx}}^H} = |\alpha|^2 \left(\mathbf{W_{rx}H}\right)^H \left(\mathbf{W_{rx}H}\right) \mathbf{W_{tx}R_u} + \lambda \mathbf{W_{tx}R_u} - \alpha^* \left(\mathbf{W_{rx}H}\right)^H \mathbf{R_u} \tag{3.118}$$

Therefore the value of $\mathbf{W_{tx}}$ which yields a singular point of the Lagrangian is:

$$\mathbf{W_{tx}} = \alpha^* \left(|\alpha|^2 \left(\mathbf{W_{rx}H}\right)^H \left(\mathbf{W_{rx}H}\right) + \lambda \mathbf{I}\right)^{-1} \left(\mathbf{W_{rx}H}\right)^H \tag{3.119}$$

For reasons that will become fully clear later, it is interesting to do the following algebraic manipulation in the previous expression:

$$\begin{aligned} \mathbf{W_{tx}} &= \alpha^* \left(|\alpha|^2 \left(\mathbf{W_{rx}H}\right)^H \left(\mathbf{W_{rx}H}\right) + \lambda \mathbf{I}\right)^{-1} \left(\mathbf{W_{rx}H}\right)^H = \\ &= \frac{\alpha^*}{|\alpha|^2} \left(\left(\mathbf{W_{rx}H}\right)^H \left(\mathbf{W_{rx}H}\right) + \frac{\lambda}{|\alpha|^2} \mathbf{I}\right)^{-1} \left(\mathbf{W_{rx}H}\right)^H \end{aligned} \tag{3.120}$$

In order to keep notation compact, we define:

$$\mathbf{F} = \left(\mathbf{W_{rx}H}\right)^H \left(\mathbf{W_{rx}H}\right) + \frac{\lambda}{|\alpha|^2} \mathbf{I} \tag{3.121}$$

Note that the matrix $\mathbf{F}$ is hermitian.

The precoding matrix $\mathbf{W_{tx}}$ can be written in terms of $\mathbf{F}$ as:

$$\mathbf{W_{tx}} = \frac{\alpha^*}{|\alpha|^2} \mathbf{F}^{-1} \left(\mathbf{W_{rx}H}\right)^H \tag{3.122}$$

However, we are far from done yet. Since we have another variable, we have to differentiate the Lagrangian also with respect to α^*. Doing so we obtain:

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathbf{W_{tx}}, \alpha, \lambda)}{\partial \alpha^*} &= \alpha \operatorname{Tr}\left(\left(\mathbf{W_{rx}HW_{tx}}\right) \mathbf{R_u} \left(\mathbf{W_{rx}HW_{tx}}\right)^H + \mathbf{W_{rx}R_n W_{rx}}^H\right) \\ &\quad - \operatorname{Tr}\left(\mathbf{R_u} \left(\mathbf{W_{rx}HW_{tx}}\right)^H\right) \end{aligned} \tag{3.123}$$

Hence, the optimal value for α is then:

$$\alpha = \frac{\operatorname{Tr}\left(\mathbf{R_u} \left(\mathbf{W_{rx}HW_{tx}}\right)^H\right)}{\operatorname{Tr}\left(\left(\mathbf{W_{rx}HW_{tx}}\right) \mathbf{R_u} \left(\mathbf{W_{rx}HW_{tx}}\right)^H + \mathbf{W_{rx}R_n W_{rx}}^H\right)} \tag{3.124}$$

One can see that the expression for α seems to be the same one we would get if we did the optimization of the AGC as a scalar Wiener filter at the receiver independently (see section 3.4.1). However, the key point is that, since the precoding matrix is actually a function of α and the value of α depends on the value of the precoding matrix, we have a coupled set of equations. The difference between the Tx-WF scheme and using a MMSE-C precoder with a scalar Wiener

filter computed independently, is precisely that the second option does not take into account the coupling and, therefore, even if the expressions for $\mathbf{W_{tx}}$ and α seem analogous, the solution won't be the same. In this way, the Tx-WF precoder is more general hence it will achieve a better performance.

To solve the coupling between equations, we will introduce:

$$\widetilde{\mathbf{W}}_{\mathbf{tx}} = \frac{|\alpha|^2}{\alpha^*}\mathbf{W_{tx}} = \mathbf{F}^{-1}\left(\mathbf{W_{rx}H}\right)^H \tag{3.125}$$

Inserting that in (3.124) yields:

$$\alpha = \frac{\frac{\alpha}{|\alpha|^2}\,\mathrm{Tr}\left(\mathbf{R_u}\left(\mathbf{W_{rx}H}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)^H\right)}{\frac{1}{|\alpha|^2}\,\mathrm{Tr}\left(\left(\mathbf{W_{rx}H}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)\mathbf{R_u}\left(\mathbf{W_{rx}H}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)^H + |\alpha|^2\mathbf{W_{rx}R_n}\mathbf{W}_{\mathbf{rx}}^H\right)} \tag{3.126}$$

Canceling common terms and rearranging we can write:

$$|\alpha|^2\,\mathrm{Tr}\left(\mathbf{W_{rx}R_n}\mathbf{W}_{\mathbf{rx}}^H\right) = \mathrm{Tr}\left(\mathbf{R_u}\left(\mathbf{W_{rx}H}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)^H\right) - \mathrm{Tr}\left(\left(\mathbf{W_{rx}H}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)\mathbf{R_u}\left(\mathbf{W_{rx}H}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)^H\right) \tag{3.127}$$

The term on the right hand side can be rewritten by doing some boring algebra:

$$\begin{aligned} A &= \mathrm{Tr}\left(\mathbf{R_u}\left(\mathbf{W_{rx}H}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)^H\right) - \mathrm{Tr}\left(\left(\mathbf{W_{rx}H}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)\mathbf{R_u}\left(\mathbf{W_{rx}H}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)^H\right) = \\ &= \mathrm{Tr}\left(\mathbf{F}^{-1}\left(\mathbf{I} - \left(\mathbf{W_{rx}H}\right)^H\left(\mathbf{W_{rx}H}\right)\mathbf{F}^{-1}\right)\left(\mathbf{W_{rx}H}\right)^H\mathbf{R_u}\left(\mathbf{W_{rx}H}\right)\right) \end{aligned} \tag{3.128}$$

It can be further developed if we introduce the unitarily diagonalized form of the matrix $\left(\mathbf{W_{rx}H}\right)^H\left(\mathbf{W_{rx}H}\right) = \mathbf{UDU}^H$. With this we can write:

$$\mathbf{F}^{-1} = \mathbf{U}\left(\mathbf{D} + \frac{\lambda}{|\alpha|^2}\mathbf{I}\right)^{-1}\mathbf{U}^H \tag{3.129}$$

And then:

$$\mathbf{I} - \left(\mathbf{W_{rx}H}\right)^H\left(\mathbf{W_{rx}H}\right)\mathbf{F}^{-1} = \mathbf{I} - \mathbf{U}\frac{\mathbf{D}}{\mathbf{D} + \frac{\lambda}{|\alpha|^2}\mathbf{I}}\mathbf{U}^H = \mathbf{U}\frac{\frac{\lambda}{|\alpha|^2}\mathbf{I}}{\mathbf{D} + \frac{\lambda}{|\alpha|^2}\mathbf{I}}\mathbf{U}^H \tag{3.130}$$

Therefore:

$$\mathbf{F}^{-1}\left(\mathbf{I} - \left(\mathbf{W_{rx}H}\right)^H\left(\mathbf{W_{rx}H}\right)\mathbf{F}^{-1}\right) = \mathbf{U}\frac{\frac{\lambda}{|\alpha|^2}\mathbf{I}}{\left(\mathbf{D} + \frac{\lambda}{|\alpha|^2}\mathbf{I}\right)^2}\mathbf{U}^H \tag{3.131}$$

So that we can finally write:

$$A = \mathrm{Tr}\left(\mathbf{U} \frac{\frac{\lambda}{|\alpha|^2}\mathbf{I}}{\left(\mathbf{D} + \frac{\lambda}{|\alpha|^2}\mathbf{I}\right)^2} \mathbf{U}^H \left(\mathbf{W_{rx}H}\right)^H \mathbf{R_u} \left(\mathbf{W_{rx}H}\right) \right) \tag{3.132}$$

For reasons that will be fully understood soon enough, we will forget about the previous expression for a moment and switch to something seemingly unrelated. In this formulation, we were considering a TPC expressed as $\mathrm{Tr}\left(\mathbf{W_{tx}R_u W_{tx}}^H\right) \leq P_{\max}$. If we develop the left hand side...

$$\begin{aligned} P_{\mathrm{tx}} &= \mathrm{Tr}\left(\mathbf{W_{tx}R_u W_{tx}}^H\right) = \frac{1}{|\alpha|^2}\mathrm{Tr}\left(\widetilde{\mathbf{W}}_{\mathbf{tx}}\mathbf{R_u}\widetilde{\mathbf{W}}_{\mathbf{tx}}^H\right) = \\ &= \frac{1}{|\alpha|^2}\mathrm{Tr}\left(\mathbf{F}^{-1}\left(\mathbf{W_{rx}H}\right)^H \mathbf{R_u}\left(\mathbf{W_{rx}H}\right)\mathbf{F}^{-1}\right) = \\ &= \frac{1}{|\alpha|^2}\mathrm{Tr}\left(\mathbf{F}^{-2}\left(\mathbf{W_{rx}H}\right)^H \mathbf{R_u}\left(\mathbf{W_{rx}H}\right)\right) = \\ &= \frac{1}{|\alpha|^2}\mathrm{Tr}\left(\mathbf{U} \frac{\mathbf{I}}{\left(\mathbf{D} + \frac{\lambda}{|\alpha|^2}\mathbf{I}\right)^2} \mathbf{U}^H \left(\mathbf{W_{rx}H}\right)^H \mathbf{R_u} \left(\mathbf{W_{rx}H}\right) \right) \end{aligned} \tag{3.133}$$

As we see, we reach a pretty surprising result: the total transmitted power and the term we called A are related by the very simple equation:

$$A = \lambda P_{\mathrm{tx}} \tag{3.134}$$

Moreover, we can infer from equation (3.127) that A is nothing but the total noise power at the output of the AGC. That is:

$$A = |\alpha|^2 \mathrm{Tr}\left(\mathbf{W_{rx}R_n W_{rx}}^H\right) \tag{3.135}$$

Therefore, it is possible to find the scalar term inside the matrix inverse for the expression of $\mathbf{W_{tx}}$ to be:

$$\frac{\lambda}{|\alpha|^2} = \frac{\mathrm{Tr}\left(\mathbf{W_{rx}R_n W_{rx}}^H\right)}{P_{\mathrm{tx}}} \tag{3.136}$$

The most sensible choice is to enforce the TPC with equality, that is, $P_{\mathrm{tx}} = P_{\max}$. Subsequently, we can write finally:

$$\widetilde{\mathbf{W}}_{\mathbf{tx}} = \left(\left(\mathbf{W_{rx}H}\right)^H \left(\mathbf{W_{rx}H}\right) + \frac{\mathrm{Tr}\left(\mathbf{W_{rx}R_n W_{rx}}^H\right)}{P_{\max}}\mathbf{I} \right)^{-1} \left(\mathbf{W_{rx}H}\right)^H \tag{3.137}$$

The key point of this formulation is that we have been able to analytically determine the value of $\widetilde{\mathbf{W}}_{\mathbf{tx}}$. In section 3.4.3, we saw that it was not possible to determine the value of the Lagrange multiplier λ in an easy way. However, thanks to the trick of including an AGC scalar

α, we have seen that we can obtain an analytical expression for the ratio $\frac{\lambda}{|\alpha|^2}$. This solves the problem of finding a scalar which is within the inverse of a sum of matrices, allowing us to get the sought solution without resorting to numerical methods. Thus, we end up avoiding the most computationally expensive step needed for computing the MMSE-C precoder.

Finally, we still need to solve for the value of α. Since we now know the value of $\widetilde{\mathbf{W}}_{\mathbf{tx}}$, we can use equation (3.133) to find:

$$|\alpha|^2 = \frac{P_{\max}}{\mathrm{Tr}\left(\left(\left(\mathbf{W_{rx}H}\right)^H\left(\mathbf{W_{rx}H}\right) + \frac{\mathrm{Tr}(\mathbf{W_{rx}R_nW_{rx}^H})}{P_{\max}}\mathbf{I}\right)^{-2}\left(\mathbf{W_{rx}H}\right)^H\mathbf{R_u}\left(\mathbf{W_{rx}H}\right)\right)} \tag{3.138}$$

Therefore, we see that only the magnitude of α matters, being the phase irrelevant for optimality. The simplest choice is then to take α as a real number, that is:

$$\alpha = \sqrt{\frac{P_{\max}}{\mathrm{Tr}\left(\left(\left(\mathbf{W_{rx}H}\right)^H\left(\mathbf{W_{rx}H}\right) + \frac{\mathrm{Tr}(\mathbf{W_{rx}R_nW_{rx}^H})}{P_{\max}}\mathbf{I}\right)^{-2}\left(\mathbf{W_{rx}H}\right)^H\mathbf{R_u}\left(\mathbf{W_{rx}H}\right)\right)}} \tag{3.139}$$

And using that value for α the precoding matrix is:

$$\mathbf{W_{tx}} = \frac{1}{\alpha}\left(\left(\mathbf{W_{rx}H}\right)^H\left(\mathbf{W_{rx}H}\right) + \frac{\mathrm{Tr}\left(\mathbf{W_{rx}R_nW_{rx}^H}\right)}{P_{\max}}\mathbf{I}\right)^{-1}\left(\mathbf{W_{rx}H}\right)^H \tag{3.140}$$

Or, in the equivalent but less computationally efficient form:

$$\mathbf{W_{tx}} = \frac{1}{\alpha}\left(\mathbf{W_{rx}H}\right)^H\left(\left(\mathbf{W_{rx}H}\right)\left(\mathbf{W_{rx}H}\right)^H + \frac{\mathrm{Tr}\left(\mathbf{W_{rx}R_nW_{rx}^H}\right)}{P_{\max}}\mathbf{I}\right)^{-1} \tag{3.141}$$

Unitarily diagonalizing $\left(\mathbf{W_{rx}H}\right)\left(\mathbf{W_{rx}H}\right)^H = \mathbf{VDV}^H$, the end-to-end MIMO system becomes:

$$\hat{\mathbf{u}} = \frac{1}{\alpha}\mathbf{V}\frac{\mathbf{D}}{\mathbf{D} + \frac{\mathrm{Tr}(\mathbf{W_{rx}R_nW_{rx}^H})}{P_{\max}}\mathbf{I}}\mathbf{V}^H\mathbf{u} + \alpha\mathbf{W_{rx}n} \tag{3.142}$$

With this, the error covariance matrix can be expressed in the following closed-form yet monstrous equation:

$$\begin{aligned}\mathbf{R_e} &= \left(\mathbf{I}-\frac{1}{\alpha}\mathbf{V}\frac{\mathbf{D}}{\mathbf{D}+\frac{\mathrm{Tr}(\mathbf{W_{rx}R_nW_{rx}^H})}{P_{\max}}\mathbf{I}}\mathbf{V}^H\right)\mathbf{R_u}\left(\mathbf{I}-\frac{1}{\alpha}\mathbf{V}\frac{\mathbf{D}}{\mathbf{D}+\frac{\mathrm{Tr}(\mathbf{W_{rx}R_nW_{rx}^H})}{P_{\max}}\mathbf{I}}\mathbf{V}^H\right)^H + \\ &+ |\alpha|^2\mathbf{W_{rx}R_nW_{rx}^H} = \\ &= \left(\mathbf{V}\frac{(\alpha-1)\mathbf{D}+\frac{\mathrm{Tr}(\mathbf{W_{rx}R_nW_{rx}^H})}{P_{\max}}\mathbf{I}}{\alpha\left(\mathbf{D}+\frac{\mathrm{Tr}(\mathbf{W_{rx}R_nW_{rx}^H})}{P_{\max}}\mathbf{I}\right)}\mathbf{V}^H\right)\mathbf{R_u}\left(\mathbf{V}\frac{(\alpha-1)\mathbf{D}+\frac{\mathrm{Tr}(\mathbf{W_{rx}R_nW_{rx}^H})}{P_{\max}}\mathbf{I}}{\alpha\left(\mathbf{D}+\frac{\mathrm{Tr}(\mathbf{W_{rx}R_nW_{rx}^H})}{P_{\max}}\mathbf{I}\right)}\mathbf{V}^H\right)^H = \\ &+ |\alpha|^2\mathbf{W_{rx}R_nW_{rx}^H} =\end{aligned} \tag{3.143}$$

3.4.5 PBPC Tx-WF

After having studied several MMSE schemes, both for transmit precoding and receive filtering, we can note that we lack a proper MMSE formulation which is feasible for a distributed multi-user MIMO scenario, mostly by the lack of MMSE filters which include PBPC in their design. In order to tackle this problem, we will try to develop, as a first step, a modification of the Tx-WF precoder which includes PBPC instead of a TPC. We will assume full coordination in the transmitter and no coordination in the receiver. Therefore, this scheme can be perfectly used in cellular coordinated MIMO within each cluster of coordinated base-stations.

In this section we will be working with the per-rows partition of $\mathbf{W_{tx}}$ defined in section 2.1. Hence, we will work with blocks $(\mathbf{W_{tx}})_j \in \mathbb{C}^{t\times Nr}\ \forall\, j = 1,\ldots,M$. Under this scheme, the PBPC can be expressed mathematically as:

$$\mathrm{Tr}\left((\mathbf{W_{tx}})_j\mathbf{R_u}(\mathbf{W_{tx}})_j^H\right) = P_{\max,\mathrm{j}} \tag{3.144}$$

We will assume a generalization of the scalar Wiener filter in the receiver discussed in section 3.4.4 by considering a per-antenna scalar Wiener filter in the receiver. Then $\mathbf{W_{rx}}$ can be expressed as a diagonal matrix $\mathbf{W_{rx}} = \mathbf{B}^{-1}$ with $\mathbf{B} = \mathrm{diag}\left(\left[b_1^1 b_1^2 \ldots b_1^r \ldots b_N^1 b_N^2 \ldots b_N^r\right]\right)$. Clearly, $\mathbf{B} \in \mathbb{C}^{Nr\times Nr}$ and so does $\mathbf{W_{rx}}$. The resulting system model is as shown in figure 3.3. Precisely, the effect of $\mathbf{W_{rx}}$ being a diagonal matrix $\mathbf{B}^{-1}$ is that the k-th data symbol estimate in the i-th user's receiver, $\hat{\mathbf{u}}_i^k$, is computed just by scaling by a factor $(b_i^k)^{-1}$ the signal received by the k-th antenna of that user, $\mathbf{y}_i^k$.

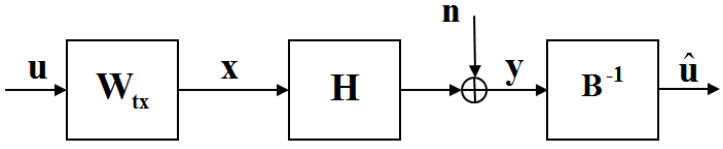


Figure 3.3: PBPC Tx-WF system model.

In order to find our precoder-receive filter pair, we want to solve the following optimization program:

$$\min_{\mathbf{W_{tx}},\mathbf{b}} \mathbb{E}\left\{\|\mathbf{u}-\hat{\mathbf{u}}\|^2\right\} \quad s.t. \quad \mathbb{E}\left\{\|(\mathbf{W_{tx}})_j\mathbf{u}\|^2\right\} = P_{\max,\mathrm{j}}\ \forall\, j=1,\ldots,M \tag{3.145}$$

Its associated Lagrangian can be written as:

$$\begin{aligned}\mathcal{L}(\mathbf{W_{tx}},\mathbf{B},\lambda_1,\ldots,\lambda_M) = &\operatorname{Tr}\left((\mathbf{B}^{-1}\mathbf{H}\mathbf{W_{tx}}-\mathbf{I})\mathbf{R_u}(\mathbf{B}^{-1}\mathbf{H}\mathbf{W_{tx}}-\mathbf{I})^H+\mathbf{B}^{-1}\mathbf{R_n}\mathbf{B}^{-H}\right)\\ &+\sum_{j=1}^{M}\lambda_j\left(\operatorname{Tr}\left((\mathbf{W_{tx}})_j\mathbf{R_u}(\mathbf{W_{tx}})_j^H\right)-P_{\max,\mathrm{j}}\right)\end{aligned} \tag{3.146}$$

We can pretty much differentiate that Lagrangian as for the case with a TPC. However, we must realize that the derivative of $f(\mathbf{W_{tx}}) = \operatorname{Tr}((\mathbf{W_{tx}})_j\mathbf{R_u}(\mathbf{W_{tx}})_j^H)$ with respect to $\mathbf{W_{tx}^H}$ is just:

$$\frac{\partial f(\mathbf{W_{tx}})}{\partial \mathbf{W_{tx}^H}} = \begin{pmatrix}\mathbf{0}_{t\times Nr}\\ \mathbf{0}_{t\times Nr}\\ \vdots\\ (\mathbf{W_{tx}})_j\\ \vdots\\ \mathbf{0}_{t\times Nr}\end{pmatrix}\mathbf{R_u} \tag{3.147}$$

We can find that by recalling that we can define the derivative of a scalar function respect of a matrix $\mathbf{X}$ as another matrix $\mathbf{D}$ such that the entry $(\mathbf{D})^{i,j}$ situated in the i-th row and the j-th column is $(\mathbf{D})^{i,j} = \frac{\partial f(\mathbf{X})}{\partial(\mathbf{X})^{i,j}}$. With that definition, $\frac{\partial \operatorname{Tr}(\mathbf{AX})}{\partial \mathbf{X}} = A^T$. For notational simplicity, we are all the time working instead with the convention $(\mathbf{D})^{i,j} = \frac{\partial f(\mathbf{X})}{\partial(\mathbf{X})^{j,i}}$ so that $\frac{\partial \operatorname{Tr}(\mathbf{AX})}{\partial \mathbf{X}} = A$. Note that since we actually use the derivative just to find stationary points, that is, to equate the derivative to the zero matrix, it's irrelevant which form we use. Using the latter convention, we find the previous derivative in a straightforward way. Note also that:

$$\frac{\partial \operatorname{Tr}((\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W_{tx}^H}))}{\partial \mathbf{W_{tx}^H}} = \sum_{j=1}^{M}\frac{\partial \operatorname{Tr}\left((\mathbf{W_{tx}})_j\mathbf{R_u}(\mathbf{W_{tx}})_j^H\right)}{\partial \mathbf{W_{tx}^H}} = \sum_{j=1}^{M}\begin{pmatrix}\mathbf{0}_{t\times Nr}\\ \mathbf{0}_{t\times Nr}\\ \vdots\\ (\mathbf{W_{tx}})_j\\ \vdots\\ \mathbf{0}_{t\times Nr}\end{pmatrix}\mathbf{R_u} = \mathbf{W_{tx}}\mathbf{R_u} \tag{3.148}$$

So we see that the previous result is consistent. By applying that, we can differentiate the Lagrangian with respect to the precoding matrix and get:

$$\frac{\partial \mathcal{L}(\mathbf{W_{tx}}, \mathbf{B}, \lambda_1, \ldots, \lambda_M)}{\partial \mathbf{W_{tx}}^H} = \mathbf{H}^H\mathbf{B}^{-2}\mathbf{H}\mathbf{W_{tx}}\mathbf{R_u} - \mathbf{H}^H\mathbf{B}^{-1}\mathbf{R_u} + \begin{pmatrix} \lambda_1(\mathbf{W_{tx}})_1 \\ \lambda_2(\mathbf{W_{tx}})_2 \\ \vdots \\ \lambda_M(\mathbf{W_{tx}})_M \end{pmatrix} \tag{3.149}$$

Equating the derivative to the zero matrix and solving yields:

$$\mathbf{W_{tx}} = (\mathbf{H}^H\mathbf{B}^{-2}\mathbf{H} + \mathbf{\Lambda})^{-1}\mathbf{H}^H\mathbf{B}^{-1} \tag{3.150}$$

Where we have introduced the matrix of dual variables $\mathbf{\Lambda}$, which can be computed as a kronecker product of a diagonal matrix with all the λ_i, that is, $\text{diag}(\lambda_1, \ldots, \lambda_M)$ and the identity matrix of size $t \times t$. Mathematically:

$$\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_M) \otimes \mathbf{I}_t \tag{3.151}$$

Now, we will take advantage of knowing the solution of $\mathbf{W_{tx}}$ to simplify the Lagrangian. We can do that as long as we differentiate the new function with respect to b_i^k taking into account that the expression of $\mathbf{W_{tx}}$ depends on the new independent variables. What we are actually doing, is to define a new function that depends only on $\mathbf{B}$ by restricting $\mathbf{W_{tx}} = \mathbf{W_{tx}}(\mathbf{B})$ to lie on a surface where the precoding matrix is optimal. That is useful because:

$$\begin{aligned} \text{Tr}\left(\mathbf{W}_{\mathbf{tx}}^H\mathbf{H}^H\mathbf{B}^{-2}\mathbf{H}\mathbf{W_{tx}}\mathbf{R_u}\right) &= \text{Tr}\left(\mathbf{W}_{\mathbf{tx}}^H(\mathbf{H}^H\mathbf{B}^{-2}\mathbf{H} + \mathbf{\Lambda})\mathbf{W_{tx}}\mathbf{R_u}\right) - \text{Tr}\left(\mathbf{W}_{\mathbf{tx}}^H\mathbf{\Lambda}\mathbf{W_{tx}}\mathbf{R_u}\right) = \\ &= \text{Tr}\left(\mathbf{B}^{-1}\mathbf{H}\mathbf{W_{tx}}\mathbf{R_u}\right) - \text{Tr}\left(\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W}_{\mathbf{tx}}^H\mathbf{\Lambda}\right) \end{aligned} \tag{3.152}$$

On the other hand, we can see that:

$$\sum_{j=1}^{M} \lambda_j(\text{Tr}\left((\mathbf{W_{tx}})_j\mathbf{R_u}(\mathbf{W_{tx}})_j^H\right)) = \text{Tr}\left(\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W}_{\mathbf{tx}}^H\mathbf{\Lambda}\right) \tag{3.153}$$

Putting all together and using the invariance of the trace function against circular shifts of its arguments and transposition we reach a very simple expression for the Lagrangian in terms of $\mathbf{B}$:

$$\mathcal{L}(\mathbf{B}, \mathbf{\Lambda}) = \text{Tr}\left(\mathbf{R_n}\mathbf{B}^{-2}\right) - \text{Tr}\left(\mathbf{B}^{-1}\mathbf{H}\mathbf{W_{tx}}(\mathbf{B}, \mathbf{\Lambda})\mathbf{R_u}\right) + \text{Tr}\left(\mathbf{R_u}\right) - \sum_{j=1}^{M} \lambda_j P_{\max,\text{j}} \tag{3.154}$$

At this point, we can use a very useful identity from the Matrix Cookbook [20], point 3.3.1 which says that for positive semidefinite, invertible matrices $\mathbf{P}$ and $\mathbf{R}$:

$$\mathbf{P}\mathbf{B}^\mathbf{H}(\mathbf{B}\mathbf{P}\mathbf{B}^\mathbf{H} + \mathbf{R})^{-1} = (\mathbf{P}^{-1} + \mathbf{B}^\mathbf{H}\mathbf{R}^{-1}\mathbf{B})^{-1}\mathbf{B}^\mathbf{H}\mathbf{R}^{-1} \tag{3.155}$$

Since both $\mathbf{\Lambda}$ and $\mathbf{B}^{-2}$ are positive semidefinite and invertible (I assume active constraints so that $\lambda_j > 0$), we can use identity (3.155) on the expression of the precoder matrix we found before, $\mathbf{W_{tx}} = (\mathbf{H}^H\mathbf{B}^{-2}\mathbf{H} + \mathbf{\Lambda})^{-1}\mathbf{H}^H\mathbf{B}^{-1}$, to obtain:

$$\mathbf{W_{tx}} = \mathbf{\Lambda}^{-1}\mathbf{H}^H(\mathbf{H}\mathbf{\Lambda}^{-1}\mathbf{H}^H + \mathbf{B^2})^{-1}\mathbf{B} \tag{3.156}$$

Plugging that in the previous expression for the Lagrangian and assuming for now that $\mathbf{R_u} = \mathbf{I}$, that is, that we have white, unit power symbols, we get:

$$\mathcal{L}(\mathbf{B}, \mathbf{\Lambda}) = \mathrm{Tr}(\mathbf{R_n}\mathbf{B}^{-2}) - \mathrm{Tr}(\mathbf{H}\mathbf{\Lambda}^{-1}\mathbf{H}^H(\mathbf{H}\mathbf{\Lambda}^{-1}\mathbf{H}^H + \mathbf{B^2})^{-1}) + \mathrm{Tr}(\mathbf{R_u}) - \sum_{j=1}^{M}\lambda_j P_{\mathrm{max,j}} \tag{3.157}$$

Now we can proceed to differentiate that function with respect to b_i^k:

$$\frac{\partial\mathcal{L}}{\partial b_i^k} = \frac{\partial(\mathrm{Tr}(\mathbf{R_n}\mathbf{B}^{-2}))}{\partial b_i^k} - \frac{\partial(\mathrm{Tr}(\mathbf{A}\mathbf{Z}^{-1}(\mathbf{B})))}{\partial b_i^k} \tag{3.158}$$

With:

$$\mathbf{A} = \mathbf{H}\mathbf{\Lambda}^{-1}\mathbf{H}^H, \quad \mathbf{Z} = \mathbf{A} + \mathbf{B^2} \tag{3.159}$$

The first term is very easy to differentiate:

$$\frac{\partial(\mathrm{Tr}(\mathbf{R_n}\mathbf{B}^{-2}))}{\partial b_i^k} = \frac{\partial\left(\sum_{i=1}^{N}\sum_{k=1}^{r}\left(\frac{\sigma_{\mathbf{n}_i^k}}{b_i^k}\right)^2\right)}{\partial b_i^k} = -2\sigma_{\mathbf{n}_i^k}^2(b_i^k)^{-3} \tag{3.160}$$

The second one, however, is pretty tricky. We need to use all the machinery of complex matrix differentiation, including the generalized chain rule, solve the problem. According to well known results, the chain rule holds for matrix derivatives if we use vectorial notation, [14]. In informal terms, the idea is to work with very long vectors instead of matrices by stacking the matrix columns and define derivatives as the Jacobian of that vector function. In [14] the authors proved that such generalized chain rule holds even for non analytic complex functions if we consider that, for each complex variable z_i, z_i and z_i^* are independent variables. Applying that, we can write:

$$\frac{\partial(\mathrm{Tr}(\mathbf{A}\mathbf{Z}^{-1}(\mathbf{B})))}{\partial b_i^k} = \frac{\partial(\mathrm{Tr}(\mathbf{A}\mathbf{Z}^{-1}(\mathbf{B})))}{\partial \mathbf{Z}}\frac{\partial \mathbf{Z}}{\partial b_i^k} \tag{3.161}$$

Looking at tables, we can find that $\frac{\partial(\mathrm{Tr}(\mathbf{A}\mathbf{Z}^{-1}(\mathbf{B})))}{\partial \mathbf{Z}} = -\mathbf{Z}^{-T}\mathbf{A}^T\mathbf{Z}^{-T}$ where we have used the matrix form notation for derivatives of scalars with respect to a matrix we discussed before. If we want to find the derivative complying with the chain rule, we need to stack up the columns of that matrix into a very long column vector and transpose it to have a row vector which is the gradient of the function.

Similarly, we have that $\mathbf{Z} = \mathbf{A} + \mathbf{B^2}$. It's straightforward to see that $\frac{\partial(\mathbf{Z})^{j,l}}{\partial b_i^k} = 0$ unless $j = l = n(i,k)$ where we introduced the function $n(i,k) = k+(i-1)r$ which allows to change from the double indexing scheme b_i^k to the standard notation with a single index. In the particular case when $j = l = n(i,k)$, then we can see that $\frac{\partial(\mathbf{Z})^{n(i,k),n(i,k)}}{\partial b_i^k} = 2b_i^k$. The whole derivative $\frac{\partial \mathbf{Z}}{\partial b_i^k}$ is then a column vector with as many rows as elements are in $\mathbf{Z}$.

Recall the matrix notation for derivatives of a scalar with respect to a matrix, that we defined as $(\mathbf{D})^{i,j} = \frac{\partial f(\mathbf{X})}{\partial(\mathbf{X})^{i,j}}$. Furthermore, let us introduce now the derivative of a matrix against a scalar as $(\mathbf{D'})^{i,j} = \frac{\partial(\mathbf{Z})^{j,i}(x)}{\partial x}$ (note the transposition of the elements in $\mathbf{Z}$ in this latter definition). Then, we have that $\frac{\partial f(x)}{\partial x} = \frac{\partial f(\mathbf{Z})}{\partial \mathbf{Z}} \frac{\partial \mathbf{Z}}{\partial x} = \mathrm{Tr}(\mathbf{D}\mathbf{D}')$ with $\mathbf{D}$ and $\mathbf{D'}$ defined as before.

Particularizing the definition of $\mathbf{D'}$ to our problem we can find that:

$$\mathbf{D'} = \frac{\partial(\mathbf{A} + \mathbf{B^2})}{\partial b_i^k} = 2b_i^k(\mathbf{J})^{n(i,k),n(i,k)} \tag{3.162}$$

Where we have introduced the single-entry matrix, $(\mathbf{J})^{i,j}$, which is defined to have all its elements with value 0 except the one at the i-th row and j-th column, which has value 1. Besides, remembering the definitions $\mathbf{A} = \mathbf{H}\mathbf{\Lambda}^{-1}\mathbf{H}^H$ and $\mathbf{Z} = \mathbf{A} + \mathbf{B^2}$ we can obtain:

$$\frac{\partial(\mathrm{Tr}(\mathbf{A}\mathbf{Z}^{-1}(\mathbf{B})))}{\partial b_i^k} = -2b_i^k\mathbf{c}_i^k \tag{3.163}$$

With $\mathbf{c} \in \mathbb{R}^{Nr}$ defined as $\mathbf{c} = \mathrm{diag}(\mathbf{C})$ and $\mathbf{C}$ given by:

$$\mathbf{C} = (\mathbf{H}\mathbf{\Lambda}^{-1}\mathbf{H}^H + \mathbf{B^2})^{-1}\mathbf{H}\mathbf{\Lambda}^{-1}\mathbf{H}^H(\mathbf{H}\mathbf{\Lambda}^{-1}\mathbf{H}^H + \mathbf{B^2})^{-1} \tag{3.164}$$

Note that we are yet once more using an indexing scheme based on an implicit partition $\mathbf{c} = \left[\mathbf{c}_1^T\mathbf{c}_2^T\ldots\mathbf{c}_N^T\right]^T$ with $\mathbf{c}_i \in \mathbb{R}^r$ and $\mathbf{c}_i^k$ being the k-th entry of $\mathbf{c}_i$. Putting all together we get:

$$\frac{\partial \mathcal{L}}{\partial b_i^k} = 2b_i^k\mathbf{c}_i^k - 2\sigma^2_{\mathbf{n}_i^k}(b_i^k)^{-3} \tag{3.165}$$

Equating that to zero yields the condition:

$$(b_i^k)^4\mathbf{c}_i^k = \sigma^2_{\mathbf{n}_i^k} \tag{3.166}$$

To make things a bit more disturbing, we can write the matrix $\mathbf{C}$ as:

$$\mathbf{C} = \mathbf{B}^{-1}\mathbf{W}_{\mathbf{tx}}^H\mathbf{\Lambda}\mathbf{W}_{\mathbf{tx}}\mathbf{B}^{-1} \tag{3.167}$$

This implies that:

$$\mathbf{c}_i^k = (b_i^k)^2((\mathbf{\Lambda}^{1/2}\mathbf{W}_{\mathbf{tx}})^H(\mathbf{\Lambda}^{1/2}\mathbf{W}_{\mathbf{tx}}))^{n(i,k),n(i,k)} \tag{3.168}$$

On the other hand, the power constraint is actually equivalent to:

$$\mathrm{Tr}\left((\mathbf{W}_{\mathbf{tx}})_j(\mathbf{W}_{\mathbf{tx}})_j^H\right) = P_{\mathrm{max,j}} \tag{3.169}$$

This can be rewritten, using the block partitioning of $\mathbf{W_{tx}}$:

$$\mathbf{W_{tx}} = \begin{pmatrix} (\mathbf{W_{tx}})_1 \\ (\mathbf{W_{tx}})_2 \\ \vdots \\ (\mathbf{W_{tx}})_M \end{pmatrix} \tag{3.170}$$

And defining the matrix:

$$\mathbf{\Psi}(\mathbf{\Lambda}, \mathbf{B}) = \mathbf{\Lambda}^{1/2}\mathbf{W_{tx}}(\mathbf{\Lambda}, \mathbf{B}) \tag{3.171}$$

Which can be expanded, because of the way we defined $\mathbf{\Lambda}$, as:

$$\mathbf{\Psi} = \mathbf{\Lambda}^{1/2}\mathbf{W_{tx}} = \begin{pmatrix} \sqrt{\lambda_1}(\mathbf{W_{tx}})_1 \\ \sqrt{\lambda_2}(\mathbf{W_{tx}})_2 \\ \vdots \\ (\sqrt{\lambda_M}(\mathbf{W_{tx}})_M \end{pmatrix} = \begin{pmatrix} (\mathbf{\Psi})_1 \\ (\mathbf{\Psi})_2 \\ \vdots \\ (\mathbf{\Psi})_M \end{pmatrix} = \begin{pmatrix} \boldsymbol{\psi}_1 & \boldsymbol{\psi}_2 & \dots & \boldsymbol{\psi}_{Nr} \end{pmatrix} \tag{3.172}$$

Where we have defined two partitions of $\mathbf{\Psi}$: one in sets of t rows, $(\mathbf{\Psi})_j \in \mathbb{C}^{t \times Nr}$, and another simply by columns, $\boldsymbol{\psi}_i \in \mathbb{C}^{Mt}$. With that, we can write:

$$\mathrm{Tr}\left((\mathbf{\Psi})_j (\mathbf{\Psi})_j^H\right) = \lambda_j P_{\mathrm{max,j}} \tag{3.173}$$

Noting also that:

$$\mathbf{c}_i^k = (b_i^k)^{-2}((\mathbf{\Lambda}^{1/2}\mathbf{W_{tx}})^H(\mathbf{\Lambda}^{1/2}\mathbf{W_{tx}}))^{n(i,k),n(i,k)} = (b_i^k)^{-2}\boldsymbol{\psi}_{n(i,k)}^H \boldsymbol{\psi}_{n(i,k)} \tag{3.174}$$

We end up with the result that solving the problem is equivalent to finding $\mathbf{\Lambda}$ and $\mathbf{B}$ so that the norm of the $n(i,k)$-th column of $\mathbf{\Psi}$ equals $\frac{\sigma^2_{\mathbf{n}_i^k}}{(b_i^k)^2}$ and the Frobenius norm of the i-th submatrix $(\mathbf{\Psi})_i$ equals $\lambda_i P_{\mathrm{max,i}}$.

Even though this formulation seems really interesting, the analytical solution is expressed in terms of an implicit matrix equation whose closed form solution has not been found yet. Besides, the fact that such equation involves the calculation of several inverses of big matrices whose elements take really small values, makes the problem really hard to tackle with standard numerical solvers. As a consequence, for now, we cannot employ this MIMO tx-rx design scheme in practice. Nevertheless, we believe we are quite close to finding a proper closed-form solution, or at least an approximation, so that it is worth pursuing further studies to finish our work and be able to make use of this promising precoder in cellular coordinated MIMO systems.

3.4.6 Interference aware Tx-WF

In the previous section, we have taken a step ahead in our quest for deriving a new MMSE filter which is suitable for distributed multi-user MIMO scenarios by introducing PBPC into the formulation of the Tx-WF made by [9] and shown in section 3.4.4 of this project.

Now, we will retreat from that advance temporarily, recovering a TPC in the formulation. On the other hand, we will try advancing in a different direction. Since this project focuses on cellular coordinated MIMO, our final goal is to develop a MMSE precoder to be used in a per-cluster basis on what we denoted as distributed multi-user MIMO scenario. This means that each base-station would use our newly developed MMSE precoding scheme coordinating with other BTSs belonging to the same cluster. However, the lack of coordination with base-stations in other clusters would imply that our system would be strongly affected by inter-cluster interference. To address this issue, we try to introduce here a concept we call interference-aware filters. The idea is that, even if we still do not allow coordination between clusters, we will introduce in the optimization problem for each cluster the interference that the transmission from within that cluster causes in the neighboring clusters. Actually, our aim is to find a compromise between within-cluster performance an inter-cluster interference. Therefore, we will reformulate now the scheme shown in section 3.4.4 to include this interference-aware concept. In the following section, we are going to combine this step and the previous one to reach our definitive precoder: an interference-aware MMSE MIMO precoding scheme with PBPC.

We will start by changing our system model to introduce awareness about the interference we are generating in the receivers belonging to other clusters. We will assume that we are designing the precoding scheme for a particular cluster with L base-stations equipped with t antennas each. Each BTS has an associated UE (User Equipment) with r receive antennas. This formulation is nothing but the one we explained in section 2.1.1. We assume that there are also $a+b$ receive antennas out of our cluster and we would like to keep the interference generated on those antennas as small as possible. To take that into account, we can write the channel matrix as:

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}} \\ \mathbf{H}_{\mathbf{c}} \\ \mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}} \end{pmatrix} \tag{3.175}$$

Where $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}} \in \mathbb{C}^{a\times Lt}$, $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}} \in \mathbb{C}^{b\times Lt}$, $\mathbf{H}_{\mathbf{c}} \in \mathbb{C}^{Lr\times Lt}$ and thus $\mathbf{H} \in \mathbb{C}^{a+b+Lr\times Lt}$. In a practical scenario, we will have full access to the value of $\mathbf{H}_{\mathbf{c}}$. However, the reader must note that having knowledge of $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}$ and $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}$ would require some sort of inter-cluster channel estimation procedure, which may be costly. Nonetheless, due to the high propagation loss of urban environments, inter-cluster interference is really severe only in the cluster-boundaries. Thus, a realistic implementation could involve each BTS knowing the channel to users in its neighboring cells, independently of whether those neighboring cells are in the same cluster or not. With that information, each cluster could construct a partial estimate of the matrices $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}$ and $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}$ to

apply this methods and the expected performance should be really close to the optimum case when $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}$ and $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}$ are fully known.

Note also that the previous partition of $\mathbf{H}$ is completely general. Depending on our notation, either $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}$ or $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}$ could be omitted since it merely amounts to a numbering convention with respect to which antennas are within-cluster antennas and which are not.

As we have been doing in most of the previous sections, we will assume a fixed and known $\mathbf{W_{rx}}$. However, it only makes sense to consider it affecting the Lr receive antennas of the cluster being considered since, by the lack of coordination, we do not have access to the value of $\mathbf{W_{rx}}$ for the other $a+b$ antennas. For notational issues, since the complexity of the notation is going to increase in this section, we will consider that the term $\mathbf{H_c}$ is actually the equivalent channel matrix as seen by the precoder of the cluster. That is, we assume that it already includes the fixed and known $\mathbf{W_{rx}}$ for the Lr antennas in the cluster. This does not change the size of the matrix since $\mathbf{W_{rx}}$ will have size $Lr \times Lr$ and the original intra-cluster channel matrix had size $Lr \times Lt$ so that the product of both still has size $Lr \times Lt$.

With all that, we can rewrite the new end-to-end system model as:

$$\begin{pmatrix} \hat{\mathbf{u}}_{\bar{c},1} \\ \hat{\mathbf{u}}_c \\ \hat{\mathbf{u}}_{\bar{c},2} \end{pmatrix} = \alpha \begin{pmatrix} \mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}} \\ \mathbf{H_c} \\ \mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}} \end{pmatrix} \mathbf{W_{tx}u} + \alpha \begin{pmatrix} \mathbf{n}_{\bar{c},1} \\ \mathbf{n}_c \\ \mathbf{n}_{\bar{c},2} \end{pmatrix} \tag{3.176}$$

As for the sizes of the elements involved in the previous model, we would have $\mathbf{u} \in \mathbb{C}^{Lr}$, $\mathbf{W_{tx}} \in \mathbb{C}^{Lt \times Lr}$, $\hat{\mathbf{u}}_{\bar{c},1}, \mathbf{n}_{\bar{c},1} \in \mathbb{C}^a$, $\hat{\mathbf{u}}_{\bar{c},1}, \mathbf{n}_{\bar{c},2} \in \mathbb{C}^b$ and, finally, $\hat{\mathbf{u}}_c, \mathbf{n}_c \in \mathbb{C}^{Lr}$. The sizes of the matrices in $\mathbf{H}$ have already been defined.

As the reader can see in the previous expression, we keep also in this new formulation the Automatic Gain Control filter implemented with a scalar α, introduced first in [9]. Moreover, that AGC is included even in the terms which are interference and, therefore, are not controlled by the cluster under design. This seems utterly inconsistent and, indeed, it is. However, as saw in section 3.4.4 when studying the Tx-WF scheme proposed in [9], the inclusion of the AGC does not affect performance. It is merely a mathematical trick which allows to solve the problem analytically, so that we can avoid resorting to numerical methods. In other words, we will assume that the AGC is present when solving the problem, because it allows us to solve it more easily. However, since a single scalar cannot affect the SNIR, any practical implementation won't have the AGC α in the antennas outside the cluster, but we still would get exactly the same performance as the one we are going to derive theoretically during this section.

In the previous equation, $\hat{\mathbf{u}}_{\bar{c},1} \in \mathbb{C}^a$ and $\hat{\mathbf{u}}_{\bar{c},2} \in \mathbb{C}^b$ represent the interference in the $a+b$ antennas outside the cluster due to the transmission originated within the cluster. On the other hand, $\hat{\mathbf{u}}_c \in \mathbb{C}^{Lr}$ is the estimate of the data symbols originated by the transmitters of the cluster computed by their associated receivers. We have also decomposed the noise vector in three portions. We will assume that those three portions are uncorrelated so that:

$$\mathbf{R_n} = \begin{pmatrix} \mathbf{R_n^{\bar{c},1}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R_n^{c}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R_n^{\bar{c},2}} \end{pmatrix} \tag{3.177}$$

A graphical representation of the current system model is depicted in figure 3.4. As we can see, the complexity has increased with respect to previous models, but the idea is still pretty much the same. Indeed, the new system model behaves as three different channels in parallel, all fed with the same signal vector $\mathbf{x} = \mathbf{W_{tx}u}$. The main novelty is the fact that two of of the three received signal vectors which arise are undesired, i.e. interference, and their norms must be minimized.

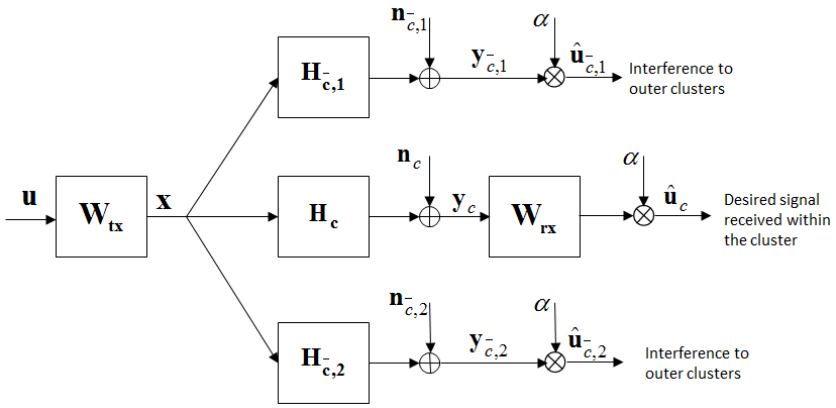


Figure 3.4: Interference-aware Tx-WF system model

The optimization problem will still be casted as a MMSE problem. However, we will not only try to minimize the mean square error between $\hat{\mathbf{u}}_c$ and $\mathbf{u}$ but, also, we include the interference terms represented by $\hat{\mathbf{u}}_{\bar{c},1}$ and $\hat{\mathbf{u}}_{\bar{c},2}$ in the cost function to try to keep them small. Hence, the resulting optimization problem becomes:

$$\min_{\mathbf{W_{tx}},\alpha} \mathbb{E}\left\{ \left\| \begin{pmatrix} \mathbf{0} \\ \mathbf{u} \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} \hat{\mathbf{u}}_{\bar{c},1} \\ \hat{\mathbf{u}}_c \\ \hat{\mathbf{u}}_{\bar{c},2} \end{pmatrix} \right\|^2 \right\} \quad s.t. \quad \mathbb{E}\left\{ \|\mathbf{W_{tx}u}\|^2 \right\} \leq P_{\max} \tag{3.178}$$

The new error correlation matrix can be found as:

$$\mathbf{R_e} = \mathbb{E}\left\{ \left(\begin{pmatrix} \mathbf{0} \\ \mathbf{u} \\ \mathbf{0} \end{pmatrix} - \alpha \begin{pmatrix} \mathbf{H_{\bar{c},1}W_{tx}} \\ \mathbf{H_cW_{tx}} \\ \mathbf{H_{\bar{c},2}W_{tx}} \end{pmatrix} \mathbf{u} \right) \left(\begin{pmatrix} \mathbf{0} \\ \mathbf{u} \\ \mathbf{0} \end{pmatrix} - \alpha \begin{pmatrix} \mathbf{H_{\bar{c},1}W_{tx}} \\ \mathbf{H_cW_{tx}} \\ \mathbf{H_{\bar{c},2}W_{tx}} \end{pmatrix} \mathbf{u} \right)^H \right\} + |\alpha|^2 \begin{pmatrix} \mathbf{R_n^{\bar{c},1}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R_n^{c}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R_n^{\bar{c},2}} \end{pmatrix} \tag{3.179}$$

Developing the previous expression, we can find that $\mathbf{R_e}$ can be written as the following partitioned matrix:

$$\mathbf{R_e} = \begin{pmatrix} f\left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}, \mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}\right) & -h\left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}, \mathbf{H_c}\right) & f\left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}, \mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}\right) \\ -h\left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}, \mathbf{H_c}\right)^H & g\left(\mathbf{H_c}\right) & -h\left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}, \mathbf{H_c}\right)^H \\ f\left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}, \mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}\right)^H & -h\left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}, \mathbf{H_c}\right) & f\left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}, \mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}\right) \end{pmatrix} + |\alpha|^2 \begin{pmatrix} \mathbf{R}_{\mathbf{n}}^{\bar{\mathbf{c}},\mathbf{1}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{\mathbf{n}}^{\mathbf{c}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{\mathbf{n}}^{\bar{\mathbf{c}},\mathbf{2}} \end{pmatrix} \tag{3.180}$$

Where we have introduced:

$$\begin{aligned} f\left(\mathbf{A}, \mathbf{B}\right) &= |\alpha|^2 \left(\mathbf{A}\mathbf{W_{tx}}\right) \mathbf{R_u} \left(\mathbf{B}\mathbf{W_{tx}}\right)^H \\ g\left(\mathbf{A}\right) &= \left(\mathbf{I} - \alpha\mathbf{A}\mathbf{W_{tx}}\right) \mathbf{R_u} \left(\mathbf{I} - \alpha\mathbf{A}\mathbf{W_{tx}}\right)^H \\ h\left(\mathbf{A}, \mathbf{B}\right) &= \alpha \left(\mathbf{A}\mathbf{W_{tx}}\right) \mathbf{R_u} \left(\mathbf{I} - \alpha\mathbf{B}\mathbf{W_{tx}}\right)^H \end{aligned} \tag{3.181}$$

We must take into account that we are only interested on $\mathrm{Tr}(\mathbf{R_e})$ for the formulation of the optimization problem we have at our hands. As a consequence, only the submatrices in the diagonal will appear in the Lagrangian:

$$\begin{aligned} \mathcal{L}(\mathbf{W_{tx}}, \alpha, \lambda) = &\mathrm{Tr}\left(\left(\mathbf{I} - \alpha\mathbf{H_c}\mathbf{W_{tx}}\right) \mathbf{R_u} \left(\mathbf{I} - \alpha\mathbf{H_c}\mathbf{W_{tx}}\right)^H\right) \\ &+ |\alpha|^2 \mathrm{Tr}\left(\left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}\mathbf{W_{tx}}\right) \mathbf{R_u} \left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}\mathbf{W_{tx}}\right)^H\right) \\ &+ |\alpha|^2 \mathrm{Tr}\left(\left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}\mathbf{W_{tx}}\right) \mathbf{R_u} \left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}\mathbf{W_{tx}}\right)^H\right) \\ &+ |\alpha|^2 \mathrm{Tr}\left(\mathbf{R_n}\right) + \lambda\left(\mathrm{Tr}\left(\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W_{tx}}^H\right) - P_{\max}\right) \end{aligned} \tag{3.182}$$

If we differentiate this new Lagrangian, we get:

$$\frac{\partial \mathcal{L}(\mathbf{W_{tx}}, \alpha, \lambda)}{\partial \mathbf{W_{tx}}^H} = |\alpha|^2 \left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}^H\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}} + \mathbf{H_c}^H\mathbf{H_c} + \mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}^H\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}\right)\mathbf{W_{tx}}\mathbf{R_u} + \lambda\mathbf{W_{tx}}\mathbf{R_u} - \alpha^*\mathbf{H_c}^H\mathbf{R_u} \tag{3.183}$$

Taking into account the definition of $\mathbf{H}$ done in equation (3.175), the previous expression can be rewritten compactly as:

$$\frac{\partial \mathcal{L}(\mathbf{W_{tx}}, \alpha, \lambda)}{\partial \mathbf{W_{tx}}^H} = |\alpha|^2\mathbf{H}^H\mathbf{H} + \lambda\mathbf{W_{tx}}\mathbf{R_u} - \alpha^*\mathbf{H_c}^H\mathbf{R_u} \tag{3.184}$$

And the final value for $\mathbf{W_{tx}}$ becomes:

$$\mathbf{W_{tx}} = \alpha^* \left(|\alpha|^2\mathbf{H}^H\mathbf{H} + \lambda\mathbf{I}\right)^{-1} \mathbf{H_c}^H \tag{3.185}$$

Just as we did in section 3.4.4 we can take $|\alpha|^2$ out of the matrix inverse to get:

$$\mathbf{W_{tx}} = \frac{\alpha^*}{|\alpha|^2} \left(\mathbf{H}^H\mathbf{H} + \frac{\lambda}{|\alpha|^2}\mathbf{I}\right)^{-1} \mathbf{H_c}^H \tag{3.186}$$

It is very interesting to analyze the differences between equation (3.186) and equation (3.120). Ignoring the fact that we included $\mathbf{W_{rx}}$ inside $\mathbf{H}$ now for notational issues, both expressions seem to be almost the same. Nonetheless, in the interference-aware formulation the matrix inverse is postmultiplied by $\mathbf{H_c}^H$ and not by the whole $\mathbf{H}$, as it is done in the regular Tx-WF.

A possible interpretation of this fact follows from noticing that Wiener filters are somehow a mixture between matched filters and zero forcing filters. Indeed, the matrix postmultiplying the inverse acts as a matched filter. Then, it makes sense that now only $\mathbf{H_c}^H$ appears, since the other blocks of $\mathbf{H}$ represent interference. In other words, the interference-aware MMSE filter is "matched" only to the intra-cluster channel.

On the other hand, the fact that $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}^H\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}$ and $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}^H\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}$ are included within the matrix inverse, actually means that the precoder treats interference as if it was noise. MMSE precoders do not include $\mathbf{R_n}$ in the formulation, which is due to the fact that they cannot act on the noise. However, MMSE receive filters do, and the way they carry it out is by including the noise autocorrelation matrix inside the matrix inverse as an additive term. This interference-aware precoder still cannot act on $\mathbf{R_n}$. Nevertheless, it can act on the interference it generates and, as we see, it does it by treating that spurious transmission just as if it was noise.

Now we have reached this point our task will be, just as in the derivation of the Tx-WF, trying to compute the value of the scalar $\frac{\lambda}{|\alpha|^2}$. Indeed, we will take exactly the very same steps, only changing slightly the expressions involved to take into account the new value of $\mathbf{W_{tx}}$.

Again we define $\mathbf{F}$ to keep notation compact as:

$$\mathbf{F} = \mathbf{H}^H\mathbf{H} + \frac{\lambda}{|\alpha|^2}\mathbf{I} \tag{3.187}$$

Just like for the Tx-WF formulation, $\mathbf{F}$ is hermitian.

Then $\mathbf{W_{tx}}$ can be rewritten as:

$$\mathbf{W_{tx}} = \frac{\alpha^*}{|\alpha|^2}\mathbf{F}^{-1}\mathbf{H_c}^H \tag{3.188}$$

Now, we will differentiate with respect α^*:

$$\frac{\partial\mathcal{L}(\mathbf{W_{tx}},\alpha,\lambda)}{\partial\alpha^*} = \alpha\,\mathrm{Tr}\left((\mathbf{HW_{tx}})\,\mathbf{R_u}\,(\mathbf{HW_{tx}})^H + \mathbf{R_n}\right) - \mathrm{Tr}\left(\mathbf{R_u}\,(\mathbf{H_cW_{tx}})^H\right) \tag{3.189}$$

Therefore, the value of α for which the Lagrangian has a singular point is:

$$\alpha = \frac{\mathrm{Tr}\left(\mathbf{R_u}\,(\mathbf{H_cW_{tx}})^H\right)}{\mathrm{Tr}\left((\mathbf{HW_{tx}})\,\mathbf{R_u}\,(\mathbf{HW_{tx}})^H + \mathbf{R_n}\right)} \tag{3.190}$$

We can see that the expression is pretty much the same as in section 3.4.4. Actually, only the numerator changes, again reflecting that this precoder is only matched to the intra-cluster part of the channel matrix.

We will solve the coupling between the equations defining the value of α and $\mathbf{W_{tx}}$ just like we did when deriving the Tx-WF. We define:

$$\widetilde{\mathbf{W}}_{\mathbf{tx}} = \frac{|\alpha|^2}{\alpha^*}\mathbf{W_{tx}} = \mathbf{F}^{-1}\mathbf{H}_{\mathbf{c}}^H \tag{3.191}$$

Inserting that in (3.190), cancelling common terms and rearranging we get something that should look familiar:

$$|\alpha|^2 \operatorname{Tr}(\mathbf{R_n}) = \operatorname{Tr}\left(\mathbf{R_u}\left(\mathbf{H_c}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)^H\right) - \operatorname{Tr}\left(\left(\mathbf{H}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)\mathbf{R_u}\left(\mathbf{H}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)^H\right) \tag{3.192}$$

We develop the expression in the right hand side of the previous equation obtaining:

$$\begin{aligned} A &= \operatorname{Tr}\left(\mathbf{R_u}\left(\mathbf{H_c}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)^H\right) - \operatorname{Tr}\left(\left(\mathbf{H}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)\mathbf{R_u}\left(\mathbf{H}\widetilde{\mathbf{W}}_{\mathbf{tx}}\right)^H\right) = \\ &= \operatorname{Tr}\left(\mathbf{F}^{-1}\left(\mathbf{I} - \mathbf{H}^H\mathbf{H}\mathbf{F}^{-1}\right)\mathbf{H}_{\mathbf{c}}^H\mathbf{R_u}\mathbf{H_c}\right) \end{aligned} \tag{3.193}$$

Unitarily diagonalizing $\mathbf{H}^H\mathbf{H} = \mathbf{U}\mathbf{D}\mathbf{U}^H$:

$$\mathbf{F}^{-1} = \mathbf{U}\left(\mathbf{D} + \frac{\lambda}{|\alpha|^2}\mathbf{I}\right)^{-1}\mathbf{U}^H \tag{3.194}$$

Therefore:

$$\mathbf{I} - \mathbf{H}^H\mathbf{H}\mathbf{F}^{-1} = \mathbf{I} - \mathbf{U}\frac{\mathbf{D}}{\mathbf{D} + \frac{\lambda}{|\alpha|^2}\mathbf{I}}\mathbf{U}^H = \mathbf{U}\frac{\frac{\lambda}{|\alpha|^2}\mathbf{I}}{\mathbf{D} + \frac{\lambda}{|\alpha|^2}\mathbf{I}}\mathbf{U}^H \tag{3.195}$$

So that:

$$\mathbf{F}^{-1}\left(\mathbf{I} - \mathbf{H}^H\mathbf{H}\mathbf{F}^{-1}\right) = \mathbf{U}\frac{\frac{\lambda}{|\alpha|^2}\mathbf{I}}{\left(\mathbf{D} + \frac{\lambda}{|\alpha|^2}\mathbf{I}\right)^2}\mathbf{U}^H \tag{3.196}$$

We can finally write:

$$A = \operatorname{Tr}\left(\mathbf{U}\frac{\frac{\lambda}{|\alpha|^2}\mathbf{I}}{\left(\mathbf{D} + \frac{\lambda}{|\alpha|^2}\mathbf{I}\right)^2}\mathbf{U}^H\mathbf{H}_{\mathbf{c}}^H\mathbf{R_u}\mathbf{H_c}\right) \tag{3.197}$$

Going back to the TPC...

$$\begin{aligned} P_{\text{tx}} &= \operatorname{Tr}\left(\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W}_{\mathbf{tx}}^H\right) = \frac{1}{|\alpha|^2}\operatorname{Tr}\left(\widetilde{\mathbf{W}}_{\mathbf{tx}}\mathbf{R_u}\widetilde{\mathbf{W}}_{\mathbf{tx}}^H\right) = \frac{1}{|\alpha|^2}\operatorname{Tr}\left(\mathbf{F}^{-1}\mathbf{H}_{\mathbf{c}}^H\mathbf{R_u}\mathbf{H_c}\mathbf{F}^{-1}\right) = \\ &= \frac{1}{|\alpha|^2}\operatorname{Tr}\left(\mathbf{F}^{-2}\mathbf{H}_{\mathbf{c}}^H\mathbf{R_u}\mathbf{H_c}\right) = \frac{1}{|\alpha|^2}\operatorname{Tr}\left(\mathbf{U}\frac{\mathbf{I}}{\left(\mathbf{D} + \frac{\lambda}{|\alpha|^2}\mathbf{I}\right)^2}\mathbf{U}^H\mathbf{H}_{\mathbf{c}}^H\mathbf{R_u}\mathbf{H_c}\right) \end{aligned} \tag{3.198}$$

As a consequence, the following equations still hold after introducing in the precoder the concept of interference-awareness:

$$A = \lambda P_{\text{tx}} \tag{3.199}$$

$$A = |\alpha|^2 \operatorname{Tr}(\mathbf{R_n}) \tag{3.200}$$

$$\frac{\lambda}{|\alpha|^2} = \frac{\operatorname{Tr}(\mathbf{R_n})}{P_{\text{tx}}} \tag{3.201}$$

Enforcing the TPC with equality as we did for deriving the Tx-WF, that is, $P_{\text{tx}} = P_{\max}$ we get:

$$\widetilde{\mathbf{W}}_{\mathbf{tx}} = \left(\mathbf{H}^H\mathbf{H} + \frac{\operatorname{Tr}(\mathbf{R_n})}{P_{\max}}\mathbf{I}\right)^{-1}\mathbf{H}_{\mathbf{c}}^H \tag{3.202}$$

Luckily, we have seen that introducing interference-awareness in the formulation of the precoder did not significatively increase the difficulty for finding an analytical, closed-form expression for $\widetilde{\mathbf{W}}_{\mathbf{tx}}$. To finish with our derivations in this section, we can find the value of α by using equation (3.198):

$$|\alpha|^2 = \frac{P_{\max}}{\operatorname{Tr}\left(\left(\mathbf{H}^H\mathbf{H} + \frac{\operatorname{Tr}(\mathbf{R_n})}{P_{\max}}\mathbf{I}\right)^{-2}\mathbf{H}_{\mathbf{c}}^H\mathbf{R_u}\mathbf{H_c}\right)} \tag{3.203}$$

Again, we can simply take α as a real number without loss of optimality. Then:

$$\alpha = \sqrt{\frac{P_{\max}}{\operatorname{Tr}\left(\left(\mathbf{H}^H\mathbf{H} + \frac{\operatorname{Tr}(\mathbf{R_n})}{P_{\max}}\mathbf{I}\right)^{-2}\mathbf{H}_{\mathbf{c}}^H\mathbf{R_u}\mathbf{H_c}\right)}} \tag{3.204}$$

And using that value for α the precoding matrix becomes:

$$\mathbf{W}_{\mathbf{tx}} = \frac{1}{\alpha}\left(\mathbf{H}^H\mathbf{H} + \frac{\operatorname{Tr}(\mathbf{R_n})}{P_{\max}}\mathbf{I}\right)^{-1}\mathbf{H}_{\mathbf{c}}^H \tag{3.205}$$

Or, in the equivalent but less computationally efficient form:

$$\mathbf{W}_{\mathbf{tx}} = \frac{1}{\alpha}\mathbf{H}_{\mathbf{c}}^H\left(\mathbf{H}\mathbf{H}^H + \frac{\operatorname{Tr}(\mathbf{R_n})}{P_{\max}}\mathbf{I}\right)^{-1} \tag{3.206}$$

3.4.7 Interference aware PBPC Tx-WF

During this section, we finally present the evolved formulation of the Tx-WF summarized in section 3.4.4 by incorporating in the original design all the ideas previously presented: a PBPC which makes the precoder applicable in distributed scenarios (3.4.5) and an interference-aware design which tries to minimize inter-cluster interference (3.4.6) besides the intra-cluster MSE.

The scenario and its corresponding system model is barely the same as in section 3.4.6. We are thinking about a cellular coordinated MIMO system where the set of M cells is partitioned in S clusters of L cells. In each of those S clusters, the L BTSs are allowed to coordinate for jointly constructing a precoder $\mathbf{W_{tx}}$. In the remainder of this section, we focus precisely on the design of the precoding matrix $\mathbf{W_{tx}}$ and receive filter $\mathbf{W_{rx}}$ for an arbitrary cluster in the system. In this way, we will be working with a set of L base-stations equipped with t antennas, where each base-station services one UE equipped with r receive antennas.

In order to keep the notation simple, we have assumed that all clusters contain the same number of cells, that is, we supposed that L is constant. However, extending our discussion to cover a cluster-dependent L is trivial, as all the steps in the derivation would be the same but making the matrices more cumbersome to write.

Again, we consider that there exist $a+b$ receive antennas outside the cluster of interest. The channel can be modeled then as in equation (3.175), where the sizes of the different submatrices are the same as in section 3.4.6, that is, $\mathbf{H_{\bar{c},1}} \in \mathbb{C}^{a\times Lt}$, $\mathbf{H_{\bar{c},2}} \in \mathbb{C}^{b\times Lt}$ and $\mathbf{H_c} \in \mathbb{C}^{Lr\times Lt}$. Also, the same considerations about the feasibility of having an estimation of the inter-cluster channel submatrices $\mathbf{H_{\bar{c},1}}$ and $\mathbf{H_{\bar{c},2}}$ apply to this formulation.

On the other hand, as we did in section 3.4.5, we will use a diagonal receive filter matrix, which amounts to a per-antenna AGC. In this way, $\mathbf{W_{rx}} = \mathbf{B}^{-1}$ with $\mathbf{B} = \mathrm{diag}\,(\mathbf{b})$ and $\mathbf{b} = \left[b_1^1 b_1^2 \dots b_1^r \dots b_L^1 b_L^2 \dots b_L^r\right]^T$. Therefore, $\mathbf{W_{rx}} \in \mathbb{C}^{Lr\times Lr}$ as it was to be expected.

In this case, given the requirement of no inter-cluster coordination, it follows that $\mathbf{W_{rx}}$ only acts in the antennas within the cluster of interest. In other words, due to the lack of coordination between different clusters, we do not have access to the receive filter matrices employed by users in the neighboring clusters. Hence, we will make no assumption about them and consider only the interference caused at the input of their receive filter.

Taking that into account the end-to-end system model is:

$$\begin{pmatrix} \hat{\mathbf{u}}_{\bar{c},1} \\ \hat{\mathbf{u}}_c \\ \hat{\mathbf{u}}_{\bar{c},2} \end{pmatrix} = \begin{pmatrix} \mathbf{H_{\bar{c},1}} \\ \mathbf{W_{rx}H_c} \\ \mathbf{H_{\bar{c},2}} \end{pmatrix} \mathbf{W_{tx}u} + \begin{pmatrix} \mathbf{n}_{\bar{c},1} \\ \mathbf{W_{rx}n}_c \\ \mathbf{n}_{\bar{c},2} \end{pmatrix} \tag{3.207}$$

And can be represented in block-diagram form as in figure 3.5.

Just like in section 3.4.6, $\hat{\mathbf{u}}_{\bar{c},1} \in \mathbb{C}^a$ and $\hat{\mathbf{u}}_{\bar{c},2} \in \mathbb{C}^b$ account for the interference in the $a+b$ antennas outside the cluster due to the signal transmitted from the cluster under study. Similarly, $\hat{\mathbf{u}}_c \in \mathbb{C}^{Lr}$ contains the data symbols as estimated by the users within the cluster being analized.

As far as the noise is concerned, we assume again that the three portions are uncorrelated so that:

$$\mathbf{R_n} = \begin{pmatrix} \mathbf{R_n^{\bar{c},1}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R_n^c} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R_n^{\bar{c},2}} \end{pmatrix} \tag{3.208}$$

Moreover, the colored noise at the output of the receive filters will be:

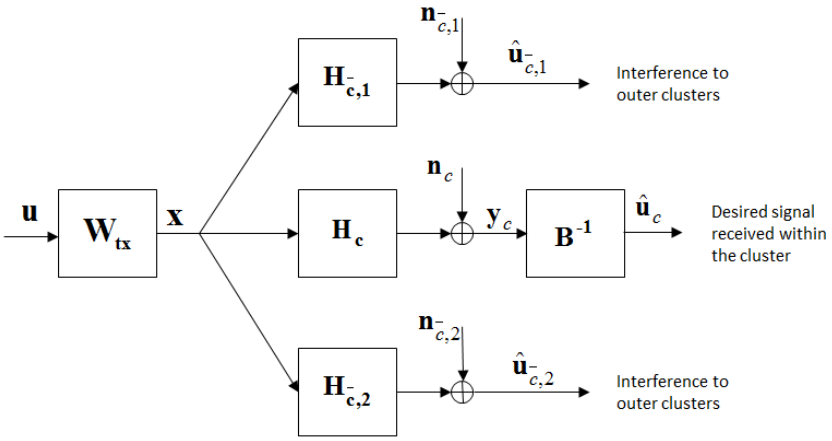


Figure 3.5: Interference Aware Tx-WF system model

$$\mathbf{R_z} = \begin{pmatrix} \mathbf{R_n^{\bar{c},1}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W_{rx}} \mathbf{R_n^c} \mathbf{W_{rx}^H} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R_n^{\bar{c},2}} \end{pmatrix} \tag{3.209}$$

Note that because of the lack of information about the receive filters outside the cluster, we assume that only the within-cluster noise is colored.

The optimization problem will be formulated in a very similar way as in section 3.4.6:

$$\min_{\mathbf{W_{tx}},\mathbf{b}} \mathbb{E}\left\{ \left\| \begin{pmatrix} \mathbf{0} \\ \mathbf{u} \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} \hat{\mathbf{u}}_{\bar{c},1} \\ \hat{\mathbf{u}}_c \\ \hat{\mathbf{u}}_{\bar{c},2} \end{pmatrix} \right\|^2 \right\} \quad s.t. \quad \mathbb{E}\left\{ \|(\mathbf{W_{tx}})_j \mathbf{u}\|^2 \right\} = P_{\max,\mathrm{j}} \; \forall j = 1, \ldots, L \tag{3.210}$$

The idea is originally the same as when we first presented the interference-aware concept: trying to include into the MSE calculation the interference we generate in neighboring clusters due to our transmission. To do so, we consider that the inter-cluster estimated symbol vectors $\hat{\mathbf{u}}_{\bar{c},1}$ and $\hat{\mathbf{u}}_{\bar{c},2}$ should ideally be $\mathbf{0}$. Then, the cost function to be optimized is obtained as the sum of the within-cluster MSE, $\mathbb{E}\left\{\|\mathbf{u} - \hat{\mathbf{u}}_c\|^2\right\}$, plus the power of the interference induced in the neighboring clusters, $\mathbb{E}\left\{\|\hat{\mathbf{u}}_{\bar{c},1}\|^2\right\} + \mathbb{E}\left\{\|\hat{\mathbf{u}}_{\bar{c},2}\|^2\right\}$.

Nevertheless, there are two differences with respect to the formulation shown in section 3.4.6.

First of all, now we are optimizing with respect to the receive filter too. In the previous formulation of the interference-aware Tx-WF we only employed an AGC in the receiver, given by a scalar factor α. Such a scalar gain is, as we discussed, irrelevant to the system's performance. However, when we included it as an optimization variable, we could reach an analytical closed-form solution for the problem. However, we are now including a different gain factor per antenna, namely, $(b_i^k)^{-1}$. Therefore, apart from the precoding matrix $\mathbf{W_{tx}}$, we now have Lr extra variables to be optimized, whereas in the previous formulation we had just 1. Those extra degrees of

freedom will allow this design to outperform its counterpart at the price of having to deal with an optimization problem which is more complex.

On the other hand, we are dealing now with a precoder designed with a distributed MIMO scenario in mind and. Therefore, we have changed the TPC in the optimization problem by a PBPC, just as we did in section 3.4.5.

The error correlation matrix can be computed basically as shown in section 3.4.6. The only difference is the omission of the AGC scalar α and the inclusion of the diagonal receive filter matrix $\mathbf{W_{rx}} = \mathbf{B}^{-1}$, only in the intra-cluster antennas:

$$\begin{aligned}\mathbf{R_e} = \mathbb{E}\left\{\left(\begin{pmatrix}\mathbf{0}\\ \mathbf{u}\\ \mathbf{0}\end{pmatrix} - \begin{pmatrix}\mathbf{H_{\bar{c},1}W_{tx}}\\ \mathbf{B^{-1}H_cW_{tx}}\\ \mathbf{H_{\bar{c},2}W_{tx}}\end{pmatrix}\mathbf{u}\right)\left(\begin{pmatrix}\mathbf{0}\\ \mathbf{u}\\ \mathbf{0}\end{pmatrix} - \begin{pmatrix}\mathbf{H_{\bar{c},1}W_{tx}}\\ \mathbf{B^{-1}H_cW_{tx}}\\ \mathbf{H_{\bar{c},2}W_{tx}}\end{pmatrix}\mathbf{u}\right)^H\right\}\\ + \begin{pmatrix}\mathbf{R_n^{\bar{c},1}} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{B^{-1}R_n^cB^{-1}} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{R_n^{\bar{c},2}}\end{pmatrix}\end{aligned} \tag{3.211}$$

Note that we have used the fact that, without loss of generality, the per-antenna AGC coefficients $(b_i^k)^{-1}$ will be real and, therefore, $\mathbf{B}^{-1}$ is a hermitian matrix. Doing a bit of algebra we can find that the interference-aware MSE is:

$$\begin{aligned}\mathrm{Tr}\left(\mathbf{R_e}\right) = &\mathrm{Tr}\left(\left(\mathbf{B^{-1}H_cW_{tx}} - \mathbf{I}\right)\mathbf{R_u}\left(\mathbf{B^{-1}H_cW_{tx}} - \mathbf{I}\right)\right)\\ &+ \mathrm{Tr}\left(\left(\mathbf{H_{\bar{c},1}W_{tx}}\right)\mathbf{R_u}\left(\mathbf{H_{\bar{c},1}W_{tx}}\right)^H\right)\\ &+ \mathrm{Tr}\left(\left(\mathbf{H_{\bar{c},2}W_{tx}}\right)\mathbf{R_u}\left(\mathbf{H_{\bar{c},2}W_{tx}}\right)^H\right)\\ &+ \mathrm{Tr}\left(\mathbf{B^{-1}R_n^cB^{-1}}\right) + \mathrm{Tr}\left(\mathbf{R_n^{\bar{c},1}}\right) + \mathrm{Tr}\left(\mathbf{R_n^{\bar{c},2}}\right)\end{aligned} \tag{3.212}$$

We can now write the Lagrangian as:

$$\begin{aligned}\mathcal{L}(\mathbf{W_{tx}}, \mathbf{B}, \mathbf{\Lambda}) = &\mathrm{Tr}\left(\left(\mathbf{B^{-1}H_cW_{tx}} - \mathbf{I}\right)\mathbf{R_u}\left(\mathbf{B^{-1}H_cW_{tx}} - \mathbf{I}\right)\right)\\ &+ \mathrm{Tr}\left(\left(\mathbf{H_{\bar{c},1}W_{tx}}\right)\mathbf{R_u}\left(\mathbf{H_{\bar{c},1}W_{tx}}\right)^H\right)\\ &+ \mathrm{Tr}\left(\left(\mathbf{H_{\bar{c},2}W_{tx}}\right)\mathbf{R_u}\left(\mathbf{H_{\bar{c},2}W_{tx}}\right)^H\right)\\ &+ \mathrm{Tr}\left(\mathbf{B^{-1}R_n^cB^{-1}}\right) + \mathrm{Tr}\left(\mathbf{R_n^{\bar{c},1}}\right) + \mathrm{Tr}\left(\mathbf{R_n^{\bar{c},2}}\right)\\ &+ \mathrm{Tr}\left(\mathbf{W_{tx}R_uW_{tx}^H\Lambda}\right) - \sum_{j=1}^{L}\lambda_j P_{\mathrm{max,j}}\end{aligned} \tag{3.213}$$

We have used that, as we saw in section 3.4.5:

$$\mathrm{Tr}\left(\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W_{tx}}^H\mathbf{\Lambda}\right) = \sum_{j=1}^{L}\lambda_j(\mathrm{Tr}\left((\mathbf{W_{tx}})_j\mathbf{R_u}(\mathbf{W_{tx}})_j^H\right)) \tag{3.214}$$

With $\mathbf{\Lambda}$ a diagonal matrix with the Lagrange multipliers, computed as:

$$\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_L) \otimes \mathbf{I}_t \tag{3.215}$$

And $\mathbf{W_{tx}}$ partitioned as:

$$\mathbf{W_{tx}} = \begin{pmatrix} (\mathbf{W_{tx}})_1 \\ (\mathbf{W_{tx}})_2 \\ \vdots \\ (\mathbf{W_{tx}})_L \end{pmatrix} \tag{3.216}$$

Where $(\mathbf{W_{tx}})_j \in \mathbb{C}^{t\times Lr}\ \forall\, j = 1, \ldots, L$. That is, the same partitioning as we used in section 3.4.5 but taking into account that now, since we are working with a single cluster in the system, there are only L cells to be considered.

We can now differentiate this Lagrangian to get:

$$\frac{\partial\mathcal{L}(\mathbf{W_{tx}}, \mathbf{B}, \mathbf{\Lambda})}{\partial\mathbf{W}_{\mathbf{tx}}^H} = \left(\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}^H\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}} + \mathbf{H}_{\mathbf{c}}^H\mathbf{B}^{-2}\mathbf{H_c} + \mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}^H\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}} + \mathbf{\Lambda}\right)\mathbf{W_{tx}}\mathbf{R_u} - \mathbf{H}_{\mathbf{c}}^H\mathbf{B}^{-1}\mathbf{R_u} \tag{3.217}$$

Equating $\frac{\partial\mathcal{L}(\mathbf{W_{tx}},\mathbf{B},\mathbf{\Lambda})}{\partial\mathbf{W}_{\mathbf{tx}}^H}$ to 0 and solving for $\mathbf{W_{tx}}$ we obtain:

$$\mathbf{W_{tx}} = \left(\mathbf{H}_{\mathbf{c}}^H\mathbf{B}^{-2}\mathbf{H_c} + \mathbf{H}_{\bar{\mathbf{c}}}^H\mathbf{H}_{\bar{\mathbf{c}}} + \mathbf{\Lambda}\right)^{-1}\mathbf{H}_{\mathbf{c}}^H\mathbf{B}^{-1} \tag{3.218}$$

Where, in order to simplify the notation, I have introduced:

$$\mathbf{H}_{\bar{\mathbf{c}}} = \begin{pmatrix} \mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}} \\ \mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}} \end{pmatrix} \tag{3.219}$$

As we can see, the expression for $\mathbf{W_{tx}}$ is actually a "mix" between equation (3.186) and equation (3.150). From the interference-aware formulation we have that the matrix inverse is postmultiplied by $\mathbf{H}_{\mathbf{c}}^H$ and not by the whole channel matrix $\mathbf{H}$, i.e., the precoder is "matched" only to the intra-cluster channel matrix. Also, $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}^H\mathbf{H}_{\bar{\mathbf{c}},\mathbf{1}}$ and $\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}^H\mathbf{H}_{\bar{\mathbf{c}},\mathbf{2}}$ are included within the matrix inverse, so that the precoder is treating interference as if it was noise. From the PBPC formulation, we have a diagonal matrix $\mathbf{\Lambda}$ included within the matrix inverse, instead of a scaled identity matrix $\lambda\mathbf{I}$. One can also notice that, since we are assuming the presence of a receive filter only in the intra-cluster antennas, the term $\mathbf{B}^{-1}$ appears always in cascade with $\mathbf{H_c}$ as $\mathbf{B}^{-1}\mathbf{H_c}$ or $\mathbf{H_c}^H\mathbf{B}^{-1}$. In other words, from the point of view of the precoder, it seems as if the intra-cluster channel matrix was $\mathbf{B}^{-1}\mathbf{H_c}$.

Just as we did in section 3.4.5, we will now substitute the value of $\mathbf{W_{tx}} = \mathbf{W_{tx}}(\mathbf{B})$ in the Lagrangian. This will simplify a lot the expression to be differentiated with respect to the per-antenna AGC coefficients b_i^k since:

$$\begin{aligned}
&\mathrm{Tr}\left(\mathbf{W}_{\mathbf{tx}}^H\left(\mathbf{H}_{\mathbf{c}}^H\mathbf{B}^{-2}\mathbf{H}_{\mathbf{c}} + \mathbf{H}_{\bar{\mathbf{c}}}^H\mathbf{H}_{\bar{\mathbf{c}}}\right)\mathbf{W_{tx}}\mathbf{R_u}\right) = \\
&= \mathrm{Tr}\left(\mathbf{W}_{\mathbf{tx}}^H\left(\mathbf{H}_{\mathbf{c}}^H\mathbf{B}^{-2}\mathbf{H}_{\mathbf{c}} + \mathbf{H}_{\bar{\mathbf{c}}}^H\mathbf{H}_{\bar{\mathbf{c}}} + \mathbf{\Lambda}\right)\mathbf{W_{tx}}\mathbf{R_u}\right) - \mathrm{Tr}\left(\mathbf{W}_{\mathbf{tx}}^H\mathbf{\Lambda}\mathbf{W_{tx}}\mathbf{R_u}\right) = \\
&= \mathrm{Tr}\left(\mathbf{B}^{-1}\mathbf{H_c}\mathbf{W_{tx}}\mathbf{R_u}\right) - \mathrm{Tr}\left(\mathbf{W_{tx}}\mathbf{R_u}\mathbf{W}_{\mathbf{tx}}^H\mathbf{\Lambda}\right)
\end{aligned} \tag{3.220}$$

With this, the Lagrangian function in terms of $\mathbf{B}$ becomes:

$$\mathcal{L}(\mathbf{B},\mathbf{\Lambda}) = \mathrm{Tr}\left(\mathbf{R}_{\mathbf{n}}^{\mathbf{c}}\mathbf{B}^{-2}\right) - \mathrm{Tr}\left(\mathbf{B}^{-1}\mathbf{H_c}\mathbf{W_{tx}}(\mathbf{B},\mathbf{\Lambda})\mathbf{R_u}\right) + \mathrm{Tr}\left(\mathbf{R_u}\right) - \sum_{j=1}^{L}\lambda_j P_{\mathrm{max,j}} \tag{3.221}$$

Since both $\mathbf{H}_{\bar{\mathbf{c}}}^H\mathbf{H}_{\bar{\mathbf{c}}} + \mathbf{\Lambda}$ and $\mathbf{B}^{-2}$ are positive semidefinite and invertible, we can use the matrix identity (3.155) on the expression $\mathbf{W_{tx}}$ to obtain:

$$\mathbf{W_{tx}} = \left(\mathbf{H}_{\bar{\mathbf{c}}}^H\mathbf{H}_{\bar{\mathbf{c}}} + \mathbf{\Lambda}\right)^{-1}\mathbf{H}_{\mathbf{c}}^H(\mathbf{H_c}\left(\mathbf{H}_{\bar{\mathbf{c}}}^H\mathbf{H}_{\bar{\mathbf{c}}} + \mathbf{\Lambda}\right)^{-1}\mathbf{H}_{\mathbf{c}}^H + \mathbf{B^2})^{-1}\mathbf{B} \tag{3.222}$$

Substituting that in the expression for the Lagrangian, and assuming that $\mathbf{R_u} = \mathbf{I}$ we get:

$$\mathcal{L}(\mathbf{B},\mathbf{\Lambda}) = \mathrm{Tr}(\mathbf{R_n}\mathbf{B}^{-2}) - \mathrm{Tr}(\mathbf{A}\mathbf{Z}^{-1}(\mathbf{B})) + \mathrm{Tr}(\mathbf{R_u}) - \sum_{j=1}^{L}\lambda_j P_{\mathrm{max,j}} \tag{3.223}$$

With:

$$\mathbf{A} = \mathbf{H_c}\left(\mathbf{H}_{\bar{\mathbf{c}}}^H\mathbf{H}_{\bar{\mathbf{c}}} + \mathbf{\Lambda}\right)^{-1}\mathbf{H}_{\mathbf{c}}^H, \quad \mathbf{Z} = \mathbf{A} + \mathbf{B^2} \tag{3.224}$$

At this point, we have to differentiate that function with respect to b_i^k. However, the reader can notice that the problem at our hands is exactly the same as in section 3.4.5. Even though the initial formulation was a bit different, as far as the differentiation with respect to b_i^k goes, we have reached an identical expression expression of the Lagrangian, the only change being the definition of the matrix $\mathbf{A}$. Therefore, we will straightforwardly use the results derived in that section to write:

$$\frac{\partial\mathcal{L}}{\partial b_i^k} = 2b_i^k\mathbf{c}_i^k - 2\sigma^2_{\mathbf{n}_i^k}(b_i^k)^{-3} \tag{3.225}$$

Where $\mathbf{c}$ is defined as a vector with the diagonal entries of the matrix $\mathbf{C} = (\mathbf{A}+\mathbf{B^2})^{-1}\mathbf{A}(\mathbf{A}+\mathbf{B^2})^{-1}$, that is, $\mathbf{c} = \mathrm{diag}(\mathbf{C})$. The indexing scheme is the same we have been using throughout this chapter. In addition to that, $\mathbf{A}$ is as in equation (3.224).

Equating the derivative to zero we get:

$$(b_i^k)^4\mathbf{c}_i^k = \sigma^2_{\mathbf{n}_i^k} \tag{3.226}$$

In other words, the solution to the PBPC Interference-Aware Tx-WF is given by a set of matrix equations analogous to that found on section 3.4.5. Introducing the interference-awareness concept merely amounts to changing the exact expression of $\mathbf{W_{tx}}$ by "matching" the precoder to the intra-cluster channel matrix and treating the inter-cluster interference as noise. It also changes the value of the matrix $\mathbf{A}$, present in the complicated implicit matrix equations with give the values of the Lagrange multipliers λ_j and the per-antenna AGC coefficients b_i^k. Therefore, we still have the same problem we had with the PBPC Tx-WF: even though the analytic design has been fulfilled, we lack a proper method of solving the implicit matrix equations which yield the desired solution.

Chapter 4

Review of clustering algorithms

4.1 Introduction to clustering

Clustering is an unsupervised machine learning technique that aims at partitioning a data set into a finite number of groups, named clusters, in such a way that objects belonging to the same group are similar to each other whereas objects belonging to different groups are dissimilar.

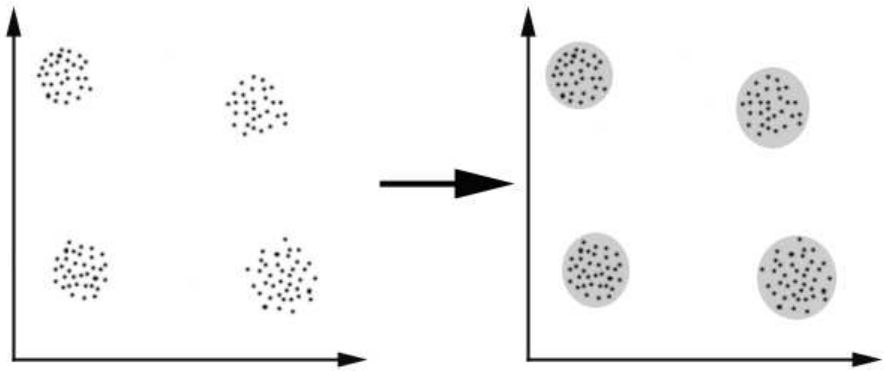


Figure 4.1: Clustering concept: The data set has been partitioned into four clusters.

Consider a data set consisting of N d-dimensional vectors $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ where $\mathbf{x}_i \in \mathbb{R}^d$. In the most general form, clustering algorithms will then find a collection of subsets of $\mathbb{X}$, $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ such that the union of all those subsets $\mathbb{S}_i$ contains all the points in the data set $\mathbb{X}$, $\bigcup_{i=1}^{k} \mathbb{S}_i = \mathbb{X}$ and no subset is empty, $\mathbb{S}_i \neq \emptyset \; \forall \, i = 1, \ldots, k$.

The reader can realize that, according to the previous definition, an object $\mathbf{x}_i$ can simultaneously belong to two distinct subsets $\mathbb{S}_j$, $\mathbb{S}_l$. When this happens, we usually say that the output is a soft partition, or that the clustering technique is overlapping or non-exclusive. One particular example is fuzzy clustering, related to the popular concept of fuzzy logic, which finds applications in several fields where ambiguity is intrinsic, such as natural language processing.

However, the most popular and well-known type of clustering algorithms are those which output a hard partition. This kind of algorithms add an extra condition on the cluster set $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$, that the clusters are mutually disjoint. Mathematically, this means than $\mathbb{S}_i \cap \mathbb{S}_j = \emptyset \; \forall \, i \neq j$. Therefore, under this setting, each object $\mathbf{x}_i$ belongs to one and only one cluster $\mathbb{S}_j$. Even though this sort of algorithms lack the flexibility of overlapping clustering, the decrease in complexity and the higher amount of knowledge about them usually pays off. Because of this, throughout this project, we will focus only on non-overlapping clustering. However, extensions of our work to consider soft-partitions constitute an interesting research line for further studies.

4.1.1 Similarity measures

The initial definition of clustering in this introduction was purposely vague, reflecting that clustering is a really broad set of techniques with many different possibilities. This is mainly because the notion of similarity (dissimilarity) needs to be measured quantitatively. In order to do that, we require to build some sort of function $f : \mathbb{X} \times \mathbb{X} \mapsto \mathbb{R}$, the similarity (dissimilarity) function, which fulfills the following properties:

1. **Positivity:** $f(\mathbf{x}, \mathbf{y}) > 0$
2. **Symmetry:** $f(\mathbf{x}, \mathbf{y}) = f(\mathbf{y}, \mathbf{x})$
3. **Maxima/minima:**
   (a) For a dissimilarity: $f(\mathbf{x}, \mathbf{x}) = 0$
   (b) For a similarity $f(\mathbf{x}, \mathbf{x}) > f(\mathbf{x}, \mathbf{y}) \quad \forall \quad \mathbf{y} \neq \mathbf{x}$

There are two considerations which are worth pointing out. First of all, we must note that, in principle, the similarity (dissimilarity) function needs to be defined only in the data set $\mathbb{X}$. However, the most popular choices for these functions tend to be continuous functions on $\mathbb{R}^d \times \mathbb{R}^d$. Also, which the properties required above, a dissimilarity does not need to be a distance metric. This is because, for a dissimilarity to be a mathematical distance metric, two extra properties are required:

1. **Coincidence axiom:** $f(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y}$
2. **Triangle inequality:** $f(\mathbf{x}, \mathbf{z}) \leq f(\mathbf{x}, \mathbf{y}) + f(\mathbf{y}, \mathbf{z})$

Again, even if the dissimilarity function does not need to fulfill those two properties, most of those typically employed in the literature do fulfill them and, therefore, are actually distance metrics over some space.

The choice between working with a similarity function or a dissimilarity function is up to the user. Indeed, we can easily convert a similarity function into a dissimilarity function by function composition with any non-increasing mapping like, for instance, $f(x) = -\log(x)$.

A few commonly used similarity functions are:

Cosine measure

The cosine measure is defined as:

$$f(\mathbf{x}, \mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2} \tag{4.1}$$

Where $\langle \bullet \rangle$ denotes the vector dot product in $\mathbb{R}^d$ and $\|\bullet\|_2$ is the Euclidean norm or L^2-norm.

This type of similarity function is particularly useful when processing text documents, like in Web search algorithms. In this scenario, we can roughly say that the documents are represented by d-dimensional vectors where each attribute contains the frequency with which a concept or word occurs in the document.

If we think about a toy example where documents are to be clustered in two categories according to the frequency of two different words, then a document is represented by a 2-D vector $\mathbf{x} = (x_1, x_2)$. We can easily see that for documents where word 1 is predominant $x_1 >> x_2$. Similarly, if word 2 is predominant, then $x_2 >> x_1$. As a consequence, documents where word 1 occurs much more frequently than word 2 are almost orthogonal to those where word 2 is much more frequent than word 2. Therefore, the cosine of both vectors would be roughly 0, so that the cosine measure would say that both documents are not similar at all, which matches our expectations. On the other hand, following the same example, it should be easy for the reader to realize that, in this scenario, it is possible to construct reasonably examples where similarity functions based on distances could give high similarity measures between documents that share no common words at all. Because of this, the cosine measure tends to perform much better in this type of scenarios.

Gaussian similarity

The Gaussian similarity function is nothing but the application of a Gaussian kernel to the observations:

$$f(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2\sigma^2}} \tag{4.2}$$

Actually, this is one example of how composing a dissimilarity function, in this case the Euclidean metric $\|\bullet\|_2$ with a non-increasing mapping, $f(x) = e^{-\frac{x^2}{2\sigma^2}}$, gives a similarity function as a result.

This similarity function is extremely useful and very interesting from a theoretical point of view given its connections to Gaussian mixtures and kernel methods. Also, they show excellent performance in a broad number of applications ranging from image segmentation to data analysis for biomedical applications.

On the other hand, some typical dissimilarity functions are:

P-norm

Probably, the most common choice for dissimilarity functions when the observations correspond to real-valued vector are those based on distances defined by the p-norm as:

$$f(\mathbf{x},\mathbf{y}) = d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_p \tag{4.3}$$

The p-norm over $\mathbb{R}^d$ is defined as:

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{d} |x_i|^p\right)^{\frac{1}{p}} \tag{4.4}$$

Special cases are:

City block metric: $p = 1 \implies \|\mathbf{x}\|_1 = \sum_{i=1}^{d} |x_i|$

Euclidean metric: $p = 2 \implies \|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{d} |x_i|^2}$

Maximum norm: $p \to \infty \implies \|\mathbf{x}\|_\infty = \max(|x_1|, |x_2|, \ldots, |x_d|)$

Many other similarity or dissimilarity functions exist in the literature. Here we have focused on those which are going to be used in this project. However, in other scenarios, like those with categorical features for instance, very different similarity or dissimilarity measures are used.

Finally, as far similarities (dissimilarities) are concerned, we must point out that all the similarities (dissimilarities) between points in the data set $\mathbb{X}$ are typically packed in a $N \times N$ matrix $\mathbf{W}$ such that $(\mathbf{W})^{i,j} = f(\mathbf{x}_i, \mathbf{x}_j)$. We will refer to that matrix as the similarity (dissimilarity) matrix of the data set.

4.1.2 Assessing the performance of a partition

When evaluating a clustering algorithm, we must first of all have a clear idea of what we might expect. The most important concept to realize is that a clustering algorithm is NOT a classification algorithm. Unlike in supervised machine learning, when along with the data set we have the set of desired outputs of the algorithm for each input observation in the training data set, clustering algorithms are unsupervised machine learning and operate in the data set alone. In other words, if we try to follow the (incorrect) analogy with classification, clustering does not only have to classify the samples, but also has to figure out how many classes are, and which points in the training data set belong to each class. The consequence of all this is simply that clustering is a much more subjective problem than classification and, therefore, we cannot expect a comparable performance in terms of the typical evaluation measures in classification like the number of misclassified samples, precision or recall,... Instead, we must consider clustering as a process which is usually used as a preprocessing step to aid other machine learning algorithms.

To illustrate the complexity, let us consider the following toy example:

Figure 4.2: Toy data set to illustrate clustering complexity

(a) A partition with two clusters

(b) A partition with four clusters

(c) A partition with six clusters

Figure 4.3: Several possible partitions for the data set shown in figure 4.2

In fact, if we conducted an experiment by asking several human being to group the data shown in figure 4.2, chances are that we will get several distinct answers. Indeed, all of the partitions in figure 4.3 seem to be reasonable choices. Then, which of the three is the correct one? The answer to that question is just that the correct partition is the one which is most useful for us in the subsequent data processing steps.

Nonetheless, up to a certain point, some partitions are obviously poorer than others. Besides, it is also possible to define measures of consistency between two partitions, which can be turned into a kind of objective cost function if one of the "partitions" is taken as a reference partition $\mathbb{P}$ which encodes the class labels opaque to the clustering algorithm. Some examples of typically employed measures to evaluate partitions of a data set are the *Rand index* and the *silhouette function*. They are useful up to a certain point, however, as we said, in the end the best way to assess the performance of a clustering algorithm is by checking its usefulness for the subsequent data processing step.

Rand Index

The Rand Index was introduced in [21] as a way to measure the consistency about two different partitions of a data set. Besides, if we take one of those partitions to be a reference partition encoding the class labels, it is possible to treat the Rand Index as an absolute measure about the "correctness" of our solution. Even though given the unsupervised nature of clustering this does not seem very useful for real applications, it is extensively used in articles as a way of comparing the performance of different clustering algorithms, hence it is interesting to introduce it here.

The Rand Index is a function of two partitions $\mathbb{A}$ and $\mathbb{B}$ of the same data set $\mathbb{X}$. It is defined as:

$$\mathrm{RI}\left(\mathbb{A}, \mathbb{B}\right) = \frac{a+d}{a+b+c+d} \tag{4.5}$$

Where a is the number of pairs of objects belonging to the same class in partition $\mathbb{A}$ and in partition $\mathbb{B}$; b is the number of pairs of objects belonging to the same class in partition $\mathbb{A}$ but to different classes in partition $\mathbb{B}$; c is the number of pairs of objects belonging to different class in partition $\mathbb{A}$ but to the same class in partition $\mathbb{B}$; and, finally, d is the number of pairs of objects belonging to different classes both in in partition $\mathbb{A}$ and in partition $\mathbb{B}$.

Both a and d are measures of consistency whereas b and c account for inconsistencies. It can be readily seen that $\mathrm{RI}\left(\mathbb{A}, \mathbb{B}\right) \in [0, 1]$ and, the bigger the Rand Index is, the higher the consistency between both partitions. A Rand Index of 0 indicates that both partitions are completely inconsistent, that is, they have absolutely nothing in common. On the contrary, a Rand Index of 1 implies both partitions to be identical.

Silhouette function

Let us consider a particular observation $\mathbf{x}_i$ in the data set $\mathbb{X}$ which has been currently assigned to cluster $\mathbb{S}_u$. Let us assume that we already obtained a dissimilarity matrix $\mathbf{W}$ according to some

dissimilarity measure or by transforming a certain similarity metric. We shall now introduce some additional notation for this section.

We define $a(i)$ as the average dissimilarity of the i-th observation $\mathbf{x}_i$ to all other samples belonging to cluster $\mathbb{S}_u$. That is:

$$a(i) = \frac{1}{|\mathbb{S}_k| - 1} \sum_{\{j|\mathbf{x}_j \in \mathbb{S}_u, j \neq i\}} (\mathbf{W})^{i,j} \tag{4.6}$$

Where $|\mathbb{S}_k|$ denotes the cardinality of the k-th cluster, that is, the number of points of the original data set in cluster $\mathbb{S}_k$.

Similarly, we introduce $d\,(i, \mathbb{S}_l)$ as the average dissimilarity between the i-th observation $\mathbf{x}_i$ and all observations belonging to cluster $\mathbb{S}_l$. Mathematically:

$$d\,(i, \mathbb{S}_l) = \frac{1}{|\mathbb{S}_l|} \sum_{\{j|\mathbf{x}_j \in \mathbb{S}_l\}} (\mathbf{W})^{i,j} \tag{4.7}$$

We can consider the "neighboring" cluster of data point $\mathbf{x}_i$ as the one which minimizes $d\,(i, \mathbb{S}_l)$. Then, the average dissimilarity between the i-th observation $\mathbf{x}_i$ and its "neighboring" cluster is defined as:

$$b(i) = \min_{1 \leq l \leq k, l \neq u} d\,(i, \mathbb{S}_l) \tag{4.8}$$

Finally, the *silhouette coefficient* associated to sample $\mathbf{x}_i$ is:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{4.9}$$

The silhouette coefficient is readily checked to fulfill $-1 \leq s(i) \leq 1$. Moreover, as the coefficient approaches 1, it means that the dissimilarity of the observation to its neighboring cluster is much greater than the dissimilarity with the cluster to which the sample belongs, i.e. a big value of $s(i)$ intuitively implies that the observation has been assigned to the "correct" cluster. On the contrary, $s(i)$ negative indicates that the dissimilarity to the neighboring cluster is smaller and we should reassign the sample to that cluster instead of the one to which it currently belongs.

Note that $s(i)$ is undefined if cluster $\mathbb{S}_l$ is a singleton, that is, $\mathbb{S}_l = \{\mathbf{x}_i\}$, because $|\mathbb{S}_k| = 1$ and $a(i)$ contains a division by 0. For this particular case, it is considered that $s(i) = 0$.

Last but not least, the silhouette function of a partition is defined as the average of all silhouette coefficients:

$$C(\mathbb{S}) = \frac{1}{N} \sum_{i=1}^{N} s(i) \tag{4.10}$$

Obviously, $C(\mathbb{S})$ also belongs to the interval $(-1, 1)$ and exhibits the same intuitive properties as the silhouette coefficient. The closer $C(\mathbb{S})$ is to unity, we can expect that the partition will be better.

4.1.3 Applications of clustering

Clustering is an essential tool for exploratory data analysis, with applications in a very wide variety of fields ranging from bioinformatics and medicine, statistics and statistical signal processing or computer science to psychology or social sciences. Clustering attempts to solve one of the fundamental steps in data analysis when people try to get a first impression on the data by looking for patterns of similar behavior or groups. As consequence, there is an increasing interest in the topic leading to the design of hundreds of different clustering algorithms customized for each particular application. Some examples of typical applications for clustering algorithms are:

Bioinformatics

One of the fields where clustering techniques have been shown to be really promising is bioinformatics. Within the big set of applications that clustering finds in bioinformatics, the most important nowadays is within the analysis of gene expression data.

Proteins are the base of most of the essential processes in life, including enzymes, transcription factors and cell machinery. Gene expression is the biochemical process which allows proteins to be made inside living organisms. A subset of the DNA (DeoxyriboNucleic Acid) is actually a "chemical reference book" which allows the cells to build protein. Indeed, a group of three nucleotides, called a "codon", codes for one of the different twenty amino acids, the build blocks of proteins.

Gene expression is the set of chemical processes which allow a cell to "read" its DNA and use it to build a particular protein. Therefore, now that the complete human genome has been sequenced and scientists have switched their attention from sequencing, to processing and understanding how genome works, the design of techniques to learn more about gene expression has become fundamental.

Gene expression is a process which comprises three different "stages": transcription, RNA (RiboNucleic Acid) processing and translation. During the transcription process, a mRNA (messenger RNA) molecule is created as a single-stranded copy of the DNA portion which codes the protein. The portions of the original DNA sequence which are not needed in the protein-making process are not copied into the mRNA molecule.

One of the latests breakthroughs in experimental molecular biology is the invention of DNA microarrays. Those allow to monitor the gene expression process at the transcript stage. DNA microarrays have a matrix structure, where rows represent genes and columns represent different samples (coming from different tissues, different developmental stages or different patients). DNA microarrays measure the relative of absolute mRNA abundance indirectly by measuring the intensity of the fluorescence of the spots on the microarray for each optical wavelength. The raw data is a set of monochrome images as the one shown in figure 4.4.

At this point, machine learning becomes the fundamental tool which allows to obtain meaningful data from the DNA microarray output. First of all, a gene-expression matrix (which will act as the data set for further data processing) has to be extracted from the raw images. We

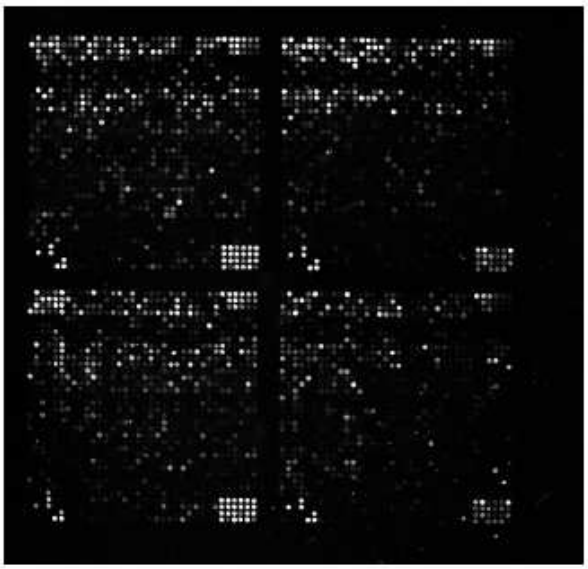


Figure 4.4: A sample image from scanning a hybridized rat microarray containing over 5000 genes. Image taken from [1].

need image processing algorithms to identify each of the spots in the image, which corresponding to a certain DNA sequence of a particular sample, determine the spot's boundaries and measure the fluorescence intensity of the spot with respect to the background intensity. The difficulty of the process greatly increases due to the existence of noise, missing values of systematic errors. Even if processing of the raw data to build the gene-expression matrix is an interesting machine learning topic by itself, since it is not related to clustering, we won't go deeper into it. As far as we are concerned now, once the raw data coming from the DNA microarray is processed, we have available a gene-expression matrix $\mathbf{W}$ of size $N - by - M$ such that N is the number of genes tested and M the number of samples. A row $\mathbf{g}_i = \left[(\mathbf{W})^{i,1}(\mathbf{W})^{i,2} \dots (\mathbf{W})^{i,M}\right]$ of the gene-expression matrix forms the expression pattern of the i-th gene, whereas a column $\mathbf{s}_j = \left[(\mathbf{W})^{1,j}(\mathbf{W})^{2,j} \dots (\mathbf{W})^{N,j}\right]^T$ represents the expression profile of the j-th sample.

Genes which exhibit similar expression patterns, referred as co-expressed genes, usually perform similar cellular functions. Moreover, the existence of a strong correlation between the expression patterns of those genes indicate co-regulation. On the other hand, clustering different samples can help us discover sub-cell types hard to identify otherwise. Therefore, we see that the application of clustering techniques allow us to increase our understanding of topics like which are the functions of each gene, how gene regulation works or the cellular processes work to name a few in a way which would be impossible without machine learning techniques.

As far as clustering is concerned, the only peculiarity of the problem is the existence of a dual data set: the data matrix $\mathbf{W}$ can be interpreted both as having the observations per rows and the features per columns (in this case each gene would be an observation and the different samples the features) or we can interpret it as having the observations per columns and the features per rows (in this case each sample would be an observation and the different

genes the features). The first case is denoted gene-based clustering whereas the second one is denoted sample-based clustering. Apart from this, most of the clustering techniques which are going to be studied in this project for a radically different application could be applied also for gene-expression data clustering with slight changes (or even no modifications at all).

The reader which desires to go further in this topic can consult [22] for a gentle introduction or [1] for a more technical example.

Marketing and Business Administration

Market segmentation is a basic and fundamental strategic marketing concept. The idea is to group people (the potential customers) according their similarity in several variables, with can range from demographics (age, gender, education, purchasing power,...), psychographics (lifestyle, beliefs, motives for buying,...) or geographics (state, city size,...). By identyfing groups of differentiated customers, the marketers can develop targeted marketing programs which address the needs and desires of each group in a specialized way, greatly increasing the customer's satisfaction.

The relation to cluster analysis is obvious since the market segmentation process is nothing but to apply a clustering algorithms where observations correspond to particular customers and the features are the variables similar to the ones shown in the previous paragraph. This data can be gathered either from different sources, like surveys of from empirical data obtained from previous products of the company. Just like with bioinformatics, most of the clustering algorithms that will be used in this project can also be applied for market segmentation with minor modifications (or no modifications at all).

Other applications

Clustering has so many applications that we cannot extend further in that matter. We have shown two important applicability niches as an examples. However, other fundamental applications can be found in many different fields like document grouping for Web searching, the recognition of communities in social networks, image processing and computer vision (image segmentation) to the processing of PET scans in medical image to differentiate between distinct types of tissue and blood. Applications for the Internet range from grouping documents or web pages in search algorithms to the recognition of communities in social networks.

Application to be developed

Along this project, we will try to develop yet another novel application for clustering. In a mobile communications environment, thinking about future technologies like 4.5G, base-station coordination along with MIMO technology will become one of the pillars that will make it possible to satisfy the throughput requirements of the standards. Up to now, the grouping of stations for coordination has always been done in a deterministic and fixed way, based on

cellular-geometry and linked to the frequency-reuse concept which was greatly used in GSM networks. However, we will show that machine learning based clustering algorithms can be developed to optimize the grouping of base-stations for coordinations in such a way that we both increase the throughput, enhancing the quality of service, and decrease the average and median cluster size, reducing the cost of the implementation of the base-station coordination process.

4.2 K-means clustering

K-means is, probably, the most well-known clustering algorithm. It is usually also the first algorithm to be explained when making an introduction to clustering since it is extremely simple and one of the most intuitive approaches that can be taken to the problem. The main idea for this algorithm goes back to 1957 and was due to Hugo Steinhays. However, the term K-means was coined by James MacQueen [23] in 1967 whereas the standard version of the algorithm was formulated by Lloyd [24], who proposed the algorithm in the context of pulse-code modulation quantization in 1957 even though the article itself was not published until 1982.

K-means is a partitional clustering algorithm which takes a data set $\mathbb{X}$ of vectors in $\mathbb{R}^d$ and a prefixed number of desired clusters k, and outputs a hard partition of the data set $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ with $\mathbb{S}_i \cap \mathbb{S}_j = \emptyset \; i \neq j$. As we shall see later, some implementations allow the existence of empty clusters, so that the effective number of clusters in that case could be less than k.

K-means uses a center-based criterion for defining the clusters, that is, each cluster $\mathbb{S}_j$ is considered to be represented by the centroid, i.e. the center of mass of the cluster. K-means then tries to find a partition such that the total sum of distances squared from each point to the centroid of the cluster they belong to is minimized. Mathematically, the cost function becomes:

$$\min_{\mathbb{S}_1,\ldots,\mathbb{S}_K} C\left(\mathbb{X}, \mathbb{S}\right) = \sum_{j=1}^{k} \sum_{i:\mathbf{x}_i \in \mathbb{S}_j} \left\|\mathbf{x}_i - \boldsymbol{\mu}_j\right\|^2 \tag{4.11}$$

Where $\boldsymbol{\mu}_j$ is the mean of the cluster, or centroid, computed as:

$$\boldsymbol{\mu}_j = \frac{1}{|\mathbb{S}_j|} \sum_{i:\mathbf{x}_i \in \mathbb{S}_j} \mathbf{x}_i \tag{4.12}$$

In order to solve this problem, K-means takes a suboptimal approach based on greedy optimization of the previous non-convex functional. The algorithm works by iteratively repeating two steps. In the first one, we assume the optimal values of the centroids $\left\{\boldsymbol{\mu}_j\right\}_{j=1}^{k}$ are known and try to find an assignment of each point $\{\mathbf{x}_i\}_{i=1}^{N}$ in the data set to a cluster $\{\mathbb{S}_j\}_{j=1}^{k}$ so that the cost function in (4.11) is minimized. Obviously, this is done by assigning $\mathbf{x}_i$ to the cluster $\mathbb{S}_j$ which satisfies $\left\|\mathbf{x}_i - \boldsymbol{\mu}_j\right\|^2 \leq \left\|\mathbf{x}_i - \boldsymbol{\mu}_k\right\|^2 \; \forall k \neq j$. It is straightforward to see that this operation is non-increasing, that is, after performing the reassignment of points in the data set to clusters

with this criterion, the worst case scenario is that the value of the cost function stays the same and, in most cases, it will decrease. The second step considers the assignation of data points to clusters fixed, and tries to optimize the value of the centroids $\{\boldsymbol{\mu}_j\}_{j=1}^{k}$ to minimize equation (4.11). We can rewrite the cost function as:

$$C(\mathbb{X}, \mathbb{S}) = \sum_{j=1}^{K} C_j(\mathbb{X}, \mathbb{S}_j) \tag{4.13}$$

Where $C_j(\mathbb{X}, \mathbb{S}_j)$ is the cost associated to the j-th cluster:

$$C_j(\mathbb{X}, \mathbb{S}_j) = \sum_{i:\mathbf{x}_i \in \mathbb{S}_j} \left\|\mathbf{x}_i - \boldsymbol{\mu}_j\right\|^2 \tag{4.14}$$

We can differentiate the cost function with respect to each of the $\{\boldsymbol{\mu}_j\}_{j=1}^{K}$ to find:

$$\nabla_{\boldsymbol{\mu}_j} C_j(\mathbb{X}, \mathbb{S}_j) = 2\left|\mathbb{S}_j\right|\boldsymbol{\mu}_j - 2\sum_{i:\mathbf{x}_i \in \mathbb{S}_j} \mathbf{x}_i \tag{4.15}$$

Setting the gradient to 0 we can find the singular point of $C_j(\mathbb{X}, \mathbb{S}_j)$ to be just the mean of all data points assigned to cluster $\mathbb{S}_j$, that is, $\boldsymbol{\mu}_j$ is as expressed in equation (4.12). Since $C_j(\mathbb{X}, \mathbb{S}_j) \geq 0$, we have that the singular point has to be a minimum, so that the value of the centroids $\{\boldsymbol{\mu}_j\}_{j=1}^{k}$ which minimize the cost function in (4.11) for a fixed assignation of data points to clusters is given by the center of mass of the points in each cluster.

K-means starts from a set of randomly generated centroids $\{\boldsymbol{\mu}_j\}_{j=1}^{k}$, and iteratively applies the two steps explained above until convergence. First, the assignment step where each data point is assigned to the cluster specified by its closest centroid. After that, centroids are recomputed according to the mean of each cluster. The process is repeated over and over until the centroids become stable.

Standard K-means can be summarized as shown in algorithm 4.1.

A toy example to illustrate the operation of K-means algorithm can be seen in figure 4.5.

We must realize that, since the optimization of the non-convex functional (4.11) is not simultaneous, taking a greedy approach, the algorithm will not generally converge to a global minimum. However, a joint optimization is a NP-hard problem, making into unfeasible in most practical cases. On the other hand, it is very easy to see prove that K-means will converge to a local minimum of (4.11) in a finite number of steps. We just need to realize that the cost function is bounded below by 0 and that the two steps of K-means, the assignment step and the centroid recalculation step are both non-increasing with respect to the cost function. Therefore, by the axiom of completeness, the algorithm will converge to a local minima. That's why the K-means algorithm is a reasonable trade-off between performance and computational complexity.

In the end, K-means generates a Voronoi tessellation in the sample space where the set of points with are closest to the j-th centroid form one cell. This guarantees that the partitions

Algorithm 4.1: Standard K-means algorithm

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \mid \mathbf{x}_i \in \mathbb{R}^d$
input : Desired number of clusters k
output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of $\mathbb{X}$

Initialize the set of centroids $\{\boldsymbol{\mu}_j\}_{j=1}^{k}$ by randomly choosing k points from $\mathbb{X}$;
while *convergence criterion not satisfied* **do**
  for $j \leftarrow 1$ **to** k **do**
    $\mathbb{S}_j = \left\{ \mathbf{x}_i \in \mathbb{X} \mid j = \min_{1 \leq n \leq k} \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|^2 \right\}$
  end
  for $j \leftarrow 1$ **to** k **do**
    $\boldsymbol{\mu}_j = \frac{1}{|\mathbb{S}_j|} \sum_{i:\mathbf{x}_i \in \mathbb{S}_j} \mathbf{x}_i$
  end
end

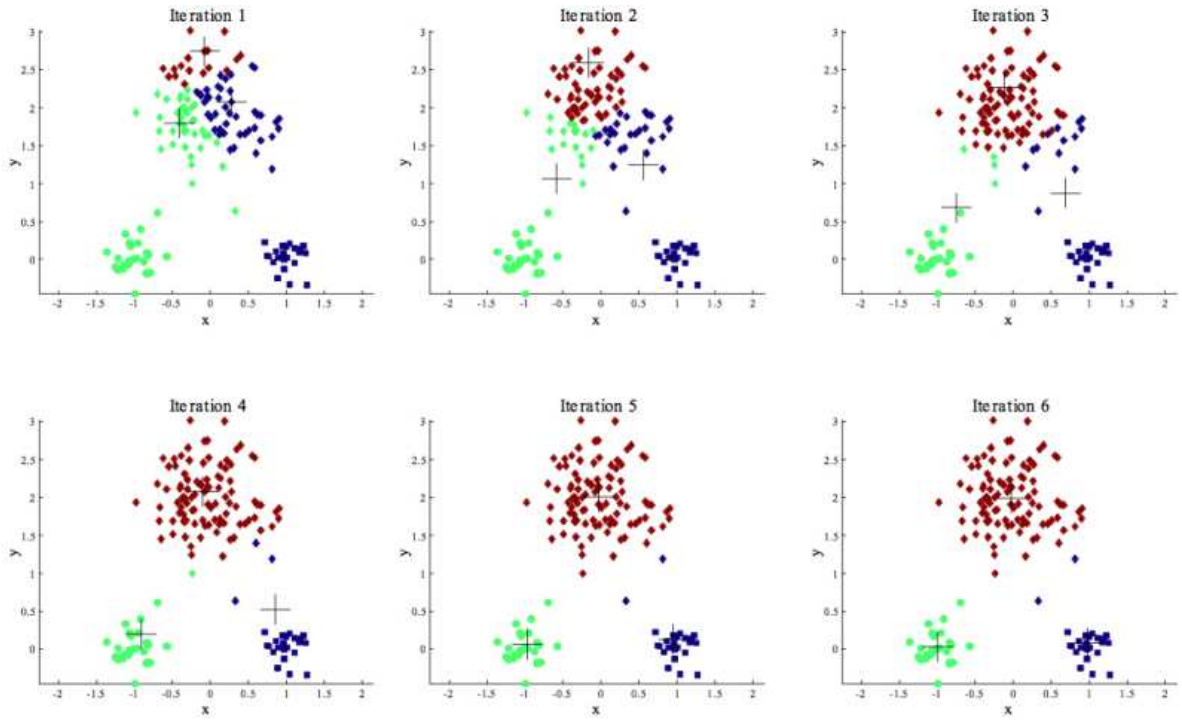


Figure 4.5: Behavior of Standard K-means algorithm for a simple data set.

generated by K-means are convex, which may be a desirable property or not depending on the shape of the data as we shall see later.

Even if it is very old already, K-means is still widely use because of its simplicity and relatively good performance. However, there exist a number of issues that make the algorithm not powerful enough in many scenarios.

Sensitivity to initialization: K-means is extremely sensitive to the random initialization of the centroids before applying the algorithm. A poor initialization, for instance, if two centroids happen to be very close to each other or some centroid is very far away from the data, can cause the appearance of empty clusters and end up degrading the performance. In figure 4.6 we can see that, for the data set of figure 4.5, an unfortunate choice for the initial centroids can lead to convergence to a local minima of poor quality. Because of this, K-means is usually run in parallel with several different initializations, and we keep the one converging to a better result (in terms of minimizing the cost function in equation (4.11)). Also, in some cases, the centroids are chosen randomly from the points in the data set, precisely to avoid the probability of getting a centroid too far away from the data samples.

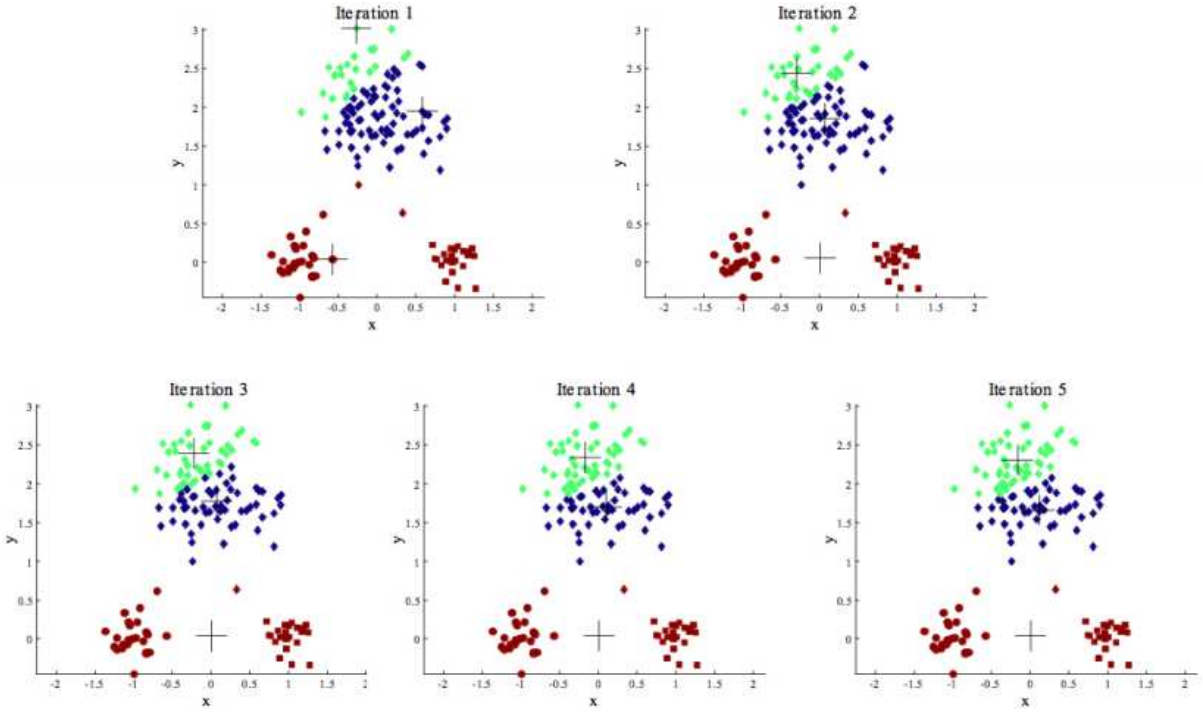


Figure 4.6: Poor choice of initial centroids leads to bad performance in K-means.

Empty clusters: As we mentioned in the previous point, it is possible that, after the assignment step, one of the clusters becomes empty. If that happens, there are two main approaches that can be taken. The first approach considers dropping the cluster, so that for the next iteration the algorithms works with $K - 1$ clusters. The other alternative is to create a new cluster to fill the gap, whose centroid is usually initialized to be the data point of the whole

data set which is furthest apart from its respective currently assigned centroid.

Number of clusters needs to be known a priori: K-means needs the number of clusters, K, as an input to operate. However, in most practical cases, determining the optimal number of clusters is already a hard problem. In practice, the best workaround consists of running the algorithm with several choices for K and keeping the best result.

Limitations: There are many scenarios which appear in everyday problems which exhibit a very bad behavior when processed with K-means. In general, K-means operates well only is the clusters are homogeneous. When the clusters differ greatly in size or density, we usually get trouble. An example can be seen in figure 4.7. Also, if the shapes are not approximately circular or when the clusters share approximately the same mean, K-means breaks down completely, since it uses a purely center-based representation of clusters. We can observe that in two different examples in figure 4.8.

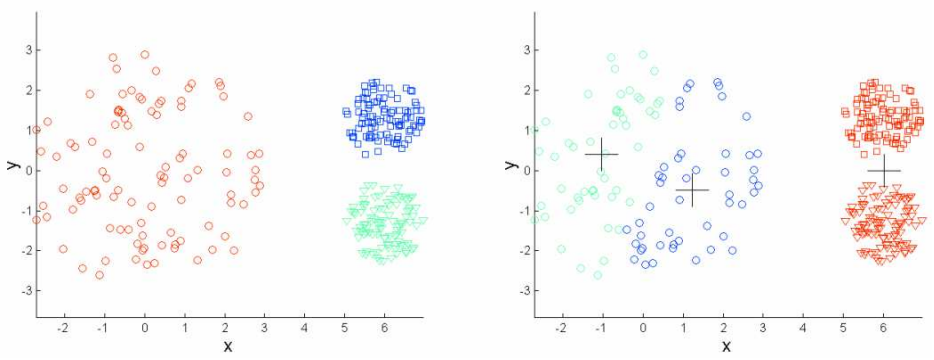
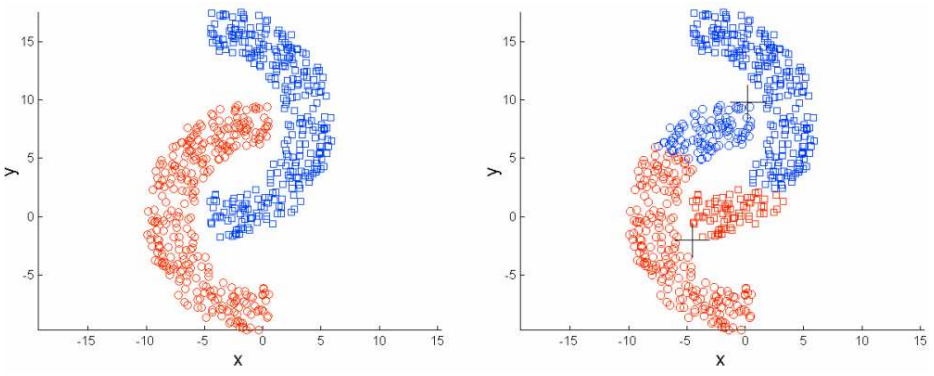


Figure 4.7: A data set with well-differentiated clusters that differ a lot in their density can be clustered poorly by K-means.

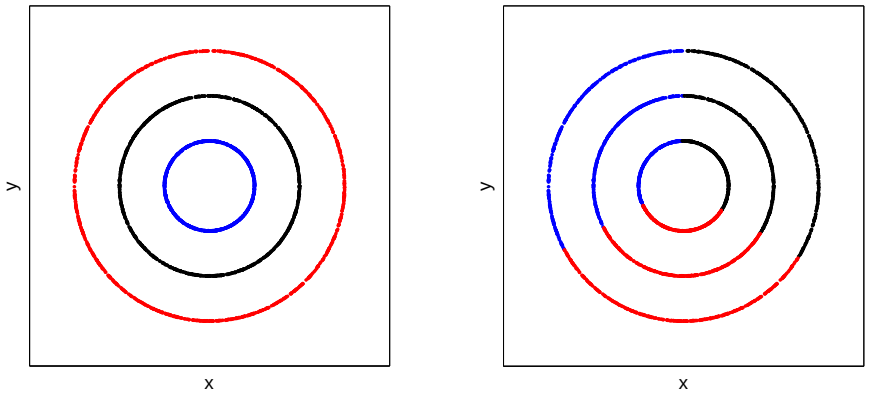
There exist a number of enhancements which solve many of the problems or limitations we just discussed. For instance, bisecting K-means increases the robustness against the initialization, evolutionary versions of K-means can find the optimal number of clusters automatically, kernel K-means performs K-means clustering in a RKHS (Reproducing Kernel Hilbert Space) of high-dimensionality, increasing the expressive power of the algorithm and making it able to deal with data of more complex shapes or there exist even fuzzy versions of K-means that perform overlapping clustering. Therefore, even if nowadays K-means is becoming progressively obsolete by the discovery of more powerful algorithms, like spectral clustering, it still offers a great variety of choice and has a wide range of applicability making it worth studying.

4.3 Hierarchical clustering

Hierarchical clustering can be considered to be a third type of clustering apart from partitional clustering and overlapping clustering. Probably, the most important peculiarity of hierarchical clustering is that it does not output a single partition but, rather, a sequence of partitions



(a) Crescents data set



(b) Rings data set

Figure 4.8: We show how K-means can exhibit a poor performance when the clusters cannot be separated by the Voronoi tesselation induced by the cluster centroids. In both examples, we shows the original data set with the "obvious" grouping on the left and the partition outputted by K-means output on the right. In 4.8(a) we have a crescents type data set where the clusters have a elongated (non-globular) shape so that the means (centroids) of the clusters are not representative. In figure 4.8(b) we have a situation which is even worse for K-means, since all the clusters share the same centroid.

$\left\{\mathbb{S}[n] = \left\{\mathbb{S}_1[n], \mathbb{S}_2[n], \ldots, \mathbb{S}[n]_{k[n]}\right\}\right\}_{n=1}^{T}$.

The name of this family of algorithms comes from the fact that they do not only seek to find an optimal partition of the data set in clusters but, also, they aim to build a hierarchy between the resulting clusters. They do so precisely by creating several different partitions over a variety of scales. In other words, the number of clusters of each of the partitions the algorithm generates, $k[n]$ ranges usually from 1, that is, all objects belonging to a single cluster, to N, which would imply that each object alone forms a cluster. Putting all the generated partitions $\{\mathbb{S}[n]\}_{n=1}^{T}$ together, we can build a tree which represents a multilevel hierarchy in such a way that each cut or level of the tree corresponding to one of the partitions $\mathbb{S}[n]$. The user must then decide which cut yields the best performance for the application at his/her hands.

There are two main groups of hierarchical clustering algorithms:

Agglomerative approaches: They are "bottom-up" algorithms. Initially, we have a partition $\mathbb{S}[0]$ where each object in the data set forms a cluster. This corresponds to starting at the lowest level in the cluster hierarchy. In each iteration, pairs of clusters are merged so that we move up in the tree until reaching the highest level, the root, where we have a partition $\mathbb{S}[T]$ in which all objects belong to the same cluster.

Divisive approaches: They are "top-down" algorithms. Their operation is exactly the opposite than that of hierarchical agglomerative clustering algorithms. The initial partition $\mathbb{S}[0]$ is now the root of the tree, treating the whole dataset as a single cluster. In each iteration, we recursively split a cluster in pieces until we reach the leaves of the tree, with a partition $\mathbb{S}[T]$ where each object is a cluster.

As we said, the set of partitions $\{\mathbb{S}[n]\}_{n=1}^{T}$ can be represented as a tree. More precisely, if we assume that clusters are merged (divided) in pairs, then we obtain a binary tree which is typically called a dendrogram. In a dendrogram, we see many interconnected upside-down U-shaped lines where each of them represents a relation between two clusters. If we "read" the U-shape "bottom-down", we see a cluster being split into two cluster and if we read it "bottom-up", we see two individual clusters being merged into a single cluster. Moreover,the height of the upside-down U-shape represents the dissimilarity (inversely proportional to the similarity) between the two clusters merged (or generated through splitting). If we plot a whole dendrogram, the leaves of the tree represent individual objects in the data set and the root would be the unique cluster containing the whole data set. Each level or cut of the tree represents one of the partitions $\mathbb{S}[n]$ generated through the iterative merging (splitting) procedure. In many cases, it is impossible to plot a complete dendrogram because of its size, so it is typical to graph only from the root up to several levels down. An example of how a typical dendrogram looks like can be found in figure 4.9.

The ability to generate dendrograms is precisely one of the main strengths of hierarchical clustering. This is because, when grouping objects into categories, in most of real life examples, there exist several perfectly valid groupings depending on the granularity. A perfect example is biological taxonomies, where all life forms are first classified into several domains. If we increase the level of detail, each domain is subdivided into several kingdoms, which are themselves divided in phyla, and so on. The standard biological taxonomic classification system considers up to eight different levels of granularity, being the most detailed one classification by species. When we use a partitional or an overlapping clustering algorithm, we can only obtain one grouping and we must set the level of detail by parametrization (if we think of K-means, that would be choosing the number of clusters K, the bigger K, the bigger the level of detail). It is clear that in many occasions, getting the whole hierarchy of groupings is extremely useful, like in the biological example.

Even though having a hierarchy of clusters is without any doubt something desirable, we will need to eventually pick one of the partitions, which actually is equivalent to cutting the dendrogram by a certain level. This can be seen more clearly in figure 4.10.

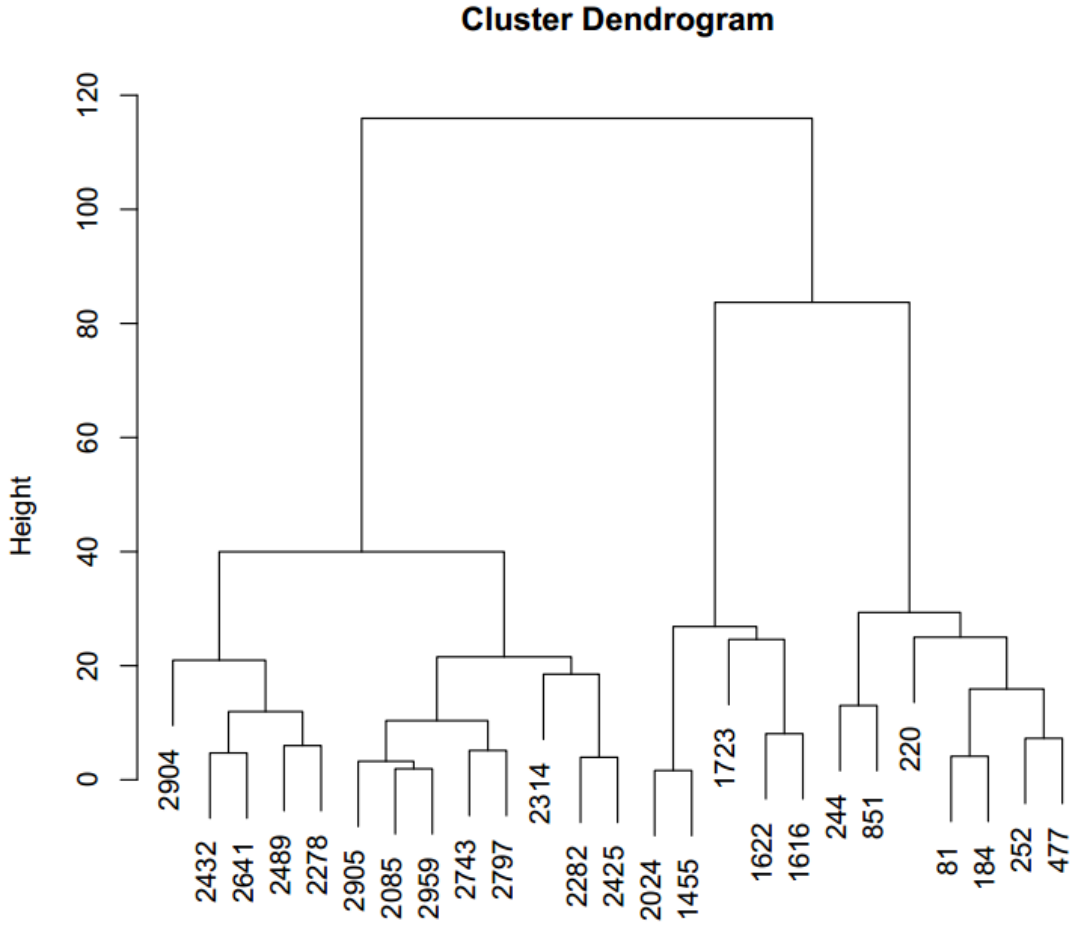


Figure 4.9: Example of dendrogram.

Bottom level: Here, each point in the dataset $\mathbb{X} = \{\mathbf{x}_i\}_{i=1}^{6}$ is a single cluster, that is:

$$\mathbb{S}[0] = \{\{\mathbf{x}_1\}, \{\mathbf{x}_2\}, \{\mathbf{x}_3\}, \{\mathbf{x}_4\}, \{\mathbf{x}_5\}, \{\mathbf{x}_6\}\} \tag{4.16}$$

First level: Points $\mathbf{x}_1$ and $\mathbf{x}_3$ have been joined in a single cluster, so that:

$$\mathbb{S}[1] = \{\{\mathbf{x}_1, \mathbf{x}_3\}, \{\mathbf{x}_2\}, \{\mathbf{x}_4\}, \{\mathbf{x}_5\}, \{\mathbf{x}_6\}\} \tag{4.17}$$

Second level: Similarly $\mathbf{x}_2$ and $\mathbf{x}_5$ get together and:

$$\mathbb{S}[2] = \{\{\mathbf{x}_1, \mathbf{x}_3\}, \{\mathbf{x}_2, \mathbf{x}_5\}, \{\mathbf{x}_4\}, \{\mathbf{x}_6\}\} \tag{4.18}$$

Third level: The cluster formed by $\mathbf{x}_2$ and $\mathbf{x}_5$ annex $\mathbf{x}_4$, yielding:

$$\mathbb{S}[3] = \{\{\mathbf{x}_1, \mathbf{x}_3\}, \{\mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_5\}, \{\mathbf{x}_6\}\} \tag{4.19}$$

Forth level: The cluster with $\mathbf{x}_1$ and $\mathbf{x}_3$ joins the cluster containing $\mathbf{x}_2$,$\mathbf{x}_4$ and $\mathbf{x}_5$:

$$\mathbb{S}[4] = \{\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5\}, \{\mathbf{x}_6\}\} \tag{4.20}$$

Top level (root): All the six original points are in the same cluster:

$$\mathbb{S}[5] = \{\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6\}\} \tag{4.21}$$

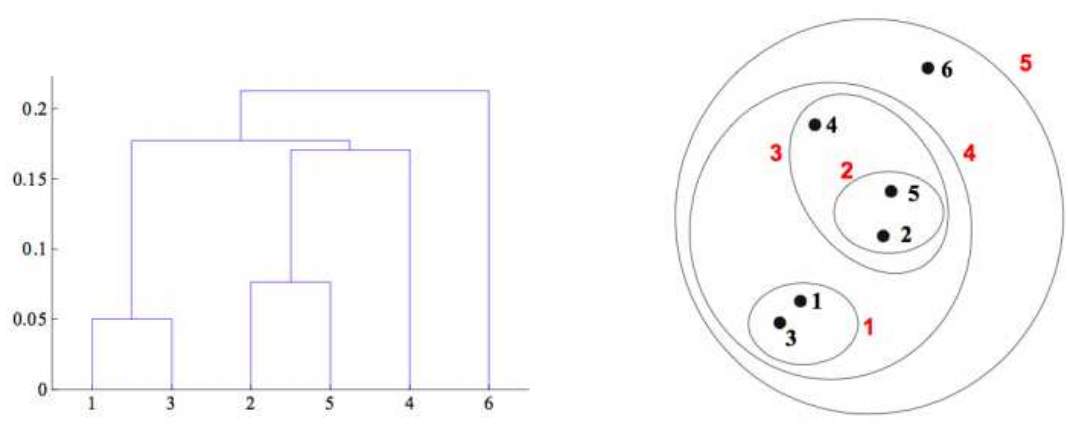


Figure 4.10: Correspondence between the different levels of a dendrogram and the distinct partitions of the data set.

It turns out that the dendrogram also provides an intuitive idea of which is the optimal level to cut the tree in order to output a single partition. Since the heights of the upside-down U-shapes is proportional to the dissimilarity between the clusters, whenever we see two clusters which merge at high values with respect to the height of the cluster unions immediately below them we can see that the merging is "weak" and we should probably cut the tree there. For instance, in figure 4.11, we see that the link leading to the root shows an abnormally big height with respect to the heights below that link. Therefore, merging those two clusters does not seem like a good idea. This is the kind of information that the dendrogram provides, which is the reason why it is such a useful tool for data analysis.

A way to make that notion precise is by computing the so called "consistency" of a link. The consistency is computed by measuring the height of the link, and the height of the links immediately below up to a prefixed depth, as shown in figure 4.12. The consistency of the i-th link is then defined as:

$$C_i = \frac{h_i - \mu_i}{\sigma_i} \tag{4.22}$$

Where h_i is the height of the i-th link, and μ_i,σ_i are the average value and standard deviation of the heights of all links considered in the calculation, respectively.

Another simpler way to decide the cut-level is simply to impose a desired number of clusters K, and cut the tree at the point which guarantees that the resulting number of clusters is at most K, but this poses the same problem as K-means and other clustering algorithms which require to know the number of clusters a priori: the choice of K may not be obvious at all.

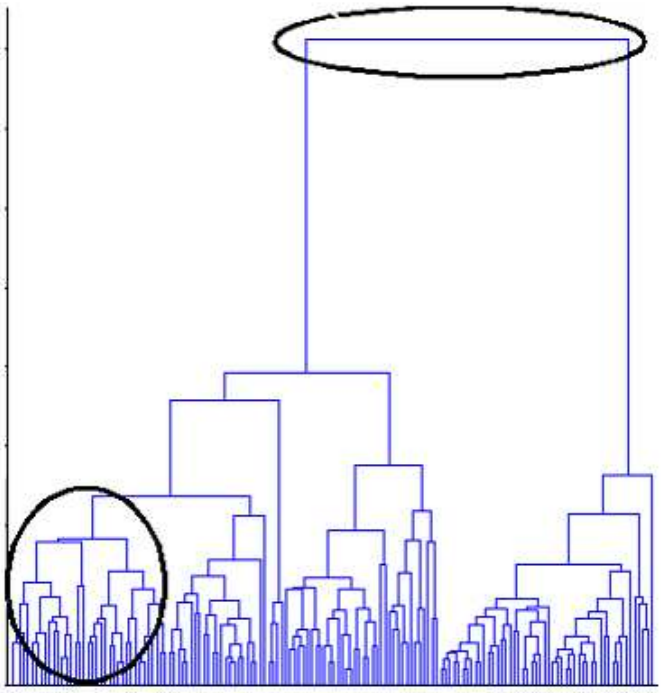


Figure 4.11: The link above has an anomalous height compared with the heights of the links below, signaling that those two clusters should not be together.

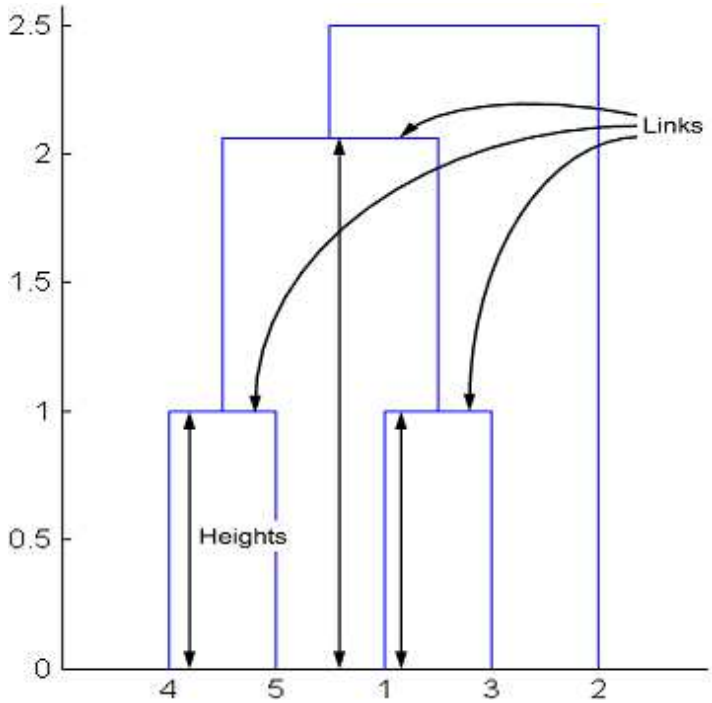


Figure 4.12: Links involved in the calculation of the consistency of the link at the second level (the one joining the cluster containing objects 4 and 5 with the cluster containing objects 1 and 3.

Hierarchical clustering is not without disadvantages either. On the one hand, most hierarchical clustering algorithms are greedy algorithms: once a decision to combine/split two clusters is made, it won't be undone. Therefore, the order of the operations may greatly affect the performance of the algorithm. Also, depending on the particular algorithm, we can experience many of the issues which affected K-means like sensitivity to outliers and noise and bad behavior with non-globular clusters or clusters with unequal densities.

After this introduction to hierarchical clustering, we will know discuss the main peculiarities of agglomerative and divisive hierarchical clustering.

4.3.1 Agglomerative hierarchical clustering

The concept of agglomerative hierarchical clustering is even more intuitive than K-means. The idea is that, at the beginning, each object to be clustered is a cluster itself. Then, we will find the two clusters which are more similar (or less dissimilar) and join them to form a bigger cluster. If repeat this process iteratively, we generate a sequence of partitions $\{\mathbb{S}[n]\}_{n=1}^{T}$ in a "bottom-up" manner until all objects are joined in a single cluster.

In the previous conceptual definition, there is a fundamental ambiguity regarding the similarities (dissimilarities). At the first step, where each cluster is initialized to be a single point in the dataset, computing the similarities (dissimilarities) between clusters is straightforward: we just need to select one of the many metrics available, as shown in the introduction of this chapter, and compute the similarity (dissimilarity) matrix $\mathbf{W}$. The first pair of objects to be merged will be that which fulfills that $(\mathbf{W})^{i,j} \geq (\mathbf{W})^{k,l}\ \forall\ (k,l) \neq (i,j)$ if we use similarities, or $(\mathbf{W})^{i,j} \leq (\mathbf{W})^{k,l}\ \forall\ (k,l) \neq (i,j)$ if we use dissimilarities instead. However, once we have done so and we have our first cluster with two objects, how do we measure the distance from that thing to the rest of points? And, more generally, how do we measure the similarity between any two clusters, with N_x and N_y points respectively? As usual, there are many different criteria on how to do define the similarity (dissimilarity) between two clusters. Those measures of similarity between clusters are usually denoted as linkages in the literature. Some of them are:

Single-linkage

Single linkage measures the dissimilarity between two clusters $\mathbb{S}_i, \mathbb{S}_j$ as the minimum dissimilarity between any two points $\mathbf{x} \in \mathbb{S}_i$, $\mathbf{y} \in \mathbb{S}_j$. If we use similarities, the argument is reverted and we seek the maximum similarity between any two points. The idea is straightforward: we take an optimistic approach in which we characterize the similarity (dissimilarity) between two clusters as the best case (in the sense of being closest) between all possible pairs of points $\mathbf{x} \in \mathbb{S}_i$, $\mathbf{y} \in \mathbb{S}_j$:

$$(\mathbf{W})^{\mathbb{S}_i,\mathbb{S}_j} = \begin{cases} \min\limits_{k:\mathbf{x}_k \in \mathbb{S}_i, l:\mathbf{x}_l \in \mathbb{S}_j} (\mathbf{W})^{k,l} & \text{if } \mathbf{W} \text{ is a dissimilarity matrix} \\ \max\limits_{k:\mathbf{x}_k \in \mathbb{S}_i, l:\mathbf{x}_l \in \mathbb{S}_j} (\mathbf{W})^{k,l} & \text{if } \mathbf{W} \text{ is a similarity matrix} \end{cases} \tag{4.23}$$

Complete-linkage

Complete linkage is the opposite of single linkage: we take the worst case scenario by characterizing the distance between two clusters as the maximum of the pairwise distances between any two points $\mathbf{x} \in \mathbb{S}_i$, $\mathbf{y} \in \mathbb{S}_j$:

$$(\mathbf{W})^{\mathbb{S}_i,\mathbb{S}_j} = \begin{cases} \max\limits_{k:\mathbf{x}_k \in \mathbb{S}_i, l:\mathbf{x}_l \in \mathbb{S}_j} (\mathbf{W})^{k,l} & \text{if } \mathbf{W} \text{ is a dissimilarity matrix} \\ \min\limits_{k:\mathbf{x}_k \in \mathbb{S}_i, l:\mathbf{x}_l \in \mathbb{S}_j} (\mathbf{W})^{k,l} & \text{if } \mathbf{W} \text{ is a similarity matrix} \end{cases} \tag{4.24}$$

Average-linkage

Just as the name indicates, average linkage evaluates the similarity (dissimilarity) between two clusters as the average of all pairwise similarities (dissimilarities) between any two points $\mathbf{x} \in \mathbb{S}_i$, $\mathbf{y} \in \mathbb{S}_j$:

$$(\mathbf{W})^{\mathbb{S}_i,\mathbb{S}_j} = \frac{1}{|\mathbb{S}_i||\mathbb{S}_j|} \sum_{k:\mathbf{x}_k \in \mathbb{S}_i} \sum_{l:\mathbf{x}_l \in \mathbb{S}_j} (\mathbf{W})^{k,l} \tag{4.25}$$

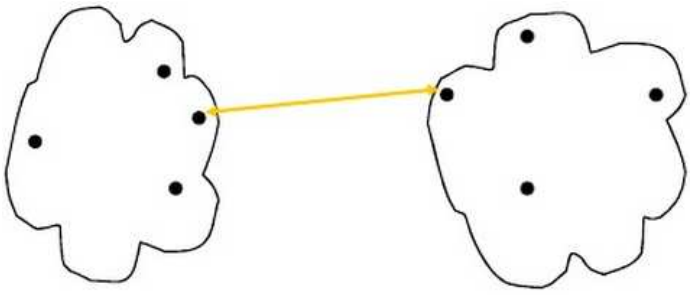
Centroid-linkage

Centroid-linkage evaluates the similarity (dissimilarity) between two clusters as the similarity (dissimilarity) between the centroids of the clusters. If $f(\mathbf{x}, \mathbf{y})$ is the similarity (dissimilarity) metric chosen, then:

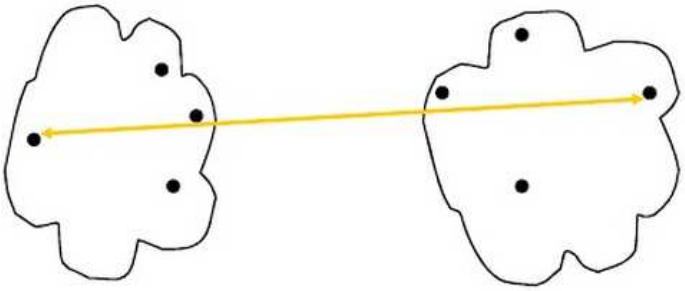
$$(\mathbf{W})^{\mathbb{S}_i,\mathbb{S}_j} = f\left(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j\right) \tag{4.26}$$

Where $\boldsymbol{\mu}_i$, $\boldsymbol{\mu}_j$ are the centroids of clusters $\mathbb{S}_i$, $\mathbb{S}_j$ respectively. Note that we can also build a median-linkage criterion if we take $\boldsymbol{\mu}_i$, $\boldsymbol{\mu}_j$ to be the medians of clusters $\mathbb{S}_i$, $\mathbb{S}_j$ instead of the centroids.

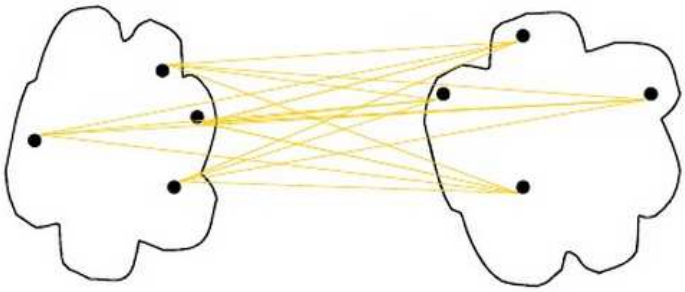
We can find a graphical representation of the different types of linkages discussed in figure 4.13. In general, single-linkage tends to handle better non-globular shapes but is quite sensitive to outliers, whereas complete-linkage is more robust against noise and outliers but usually breaks cluster which are large or have non-globular shapes. Average-linkage is a kind of compromise between the two and still tends to show difficulty to deal with non-globular clusters. Centroid (median) linkage is barely used, since it is only appropriate if the metric is the Euclidean distance (otherwise the centroid or the median are not representative points of the clusters according to our metric) and, moreover, they can produce a sequence of clusters which is non-monotonic. In this sense, monotonicity means that if we join clusters i and j at one step to form cluster a, and after that, we join them with cluster k, the dissimilarity between i and j should be smaller than the dissimilarity between the cluster a and k. This can be readily checked to be true for the first three linkages shown, but can fail with centroid-linkage or median-linkage because the centroids and medians change whenever the clusters change.



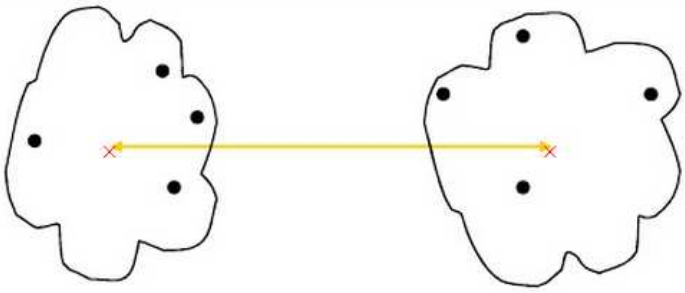
(a) Single-linkage



(b) Complete-linkage



(c) Average-linkage



(d) Centroid-linkage

Figure 4.13: Graphical illustration of several different linkages.

There exist many more linkages, like Ward's linkage, which measures the dissimilarity between two clusters as the increase in the squared error between the points and the cluster centroids (in other words, the dissimilarity equals the amount the K-means cost function degrades if we join the two clusters). Other popular linkages are based on weighted averages. As usual, the topic is far too broad to be covered in this document.

After having made clear the notion of linkage, we can write the basic agglomerative hierarchical clustering algorithm as 4.2.

Algorithm 4.2: Basic agglomerative hierarchical clustering algorithm

input : A similarity (dissimilarity) matrix $\mathbf{W}$ for the data set $\mathbb{X}$ to be clustered
input : A linkage criterion
output: A sequence of hierarchical partitions $\left\{\mathbb{S}[n] = \left\{\mathbb{S}_1[n], \mathbb{S}_2[n], \ldots, \mathbb{S}[n]_{k[n]}\right\}\right\}_{n=1}^{T}$

Initialize the first partition to $\mathbb{S}[0] \leftarrow \{\mathbb{X}\}$;
Let $\mathbf{W}_{\mathbb{S}} = \mathbf{W}$;
$n \leftarrow 0$;
while *number of clusters in* $\mathbb{S}[n]$ *greater than* 1 **do**
    Find the two clusters $\mathbb{S}_i[n]$ and $\mathbb{S}_j[n]$ which are more similar (less dissimilar) as
    $(\mathbf{W}_{\mathbb{S}})^{i,j} \geq (\mathbf{W}_{\mathbb{S}})^{k,l} \; \forall \; (k,l) \neq (i,j)$ if $\mathbf{W}_{\mathbb{S}}$ is a similarity matrix
    $(\mathbf{W}_{\mathbb{S}})^{i,j} \leq (\mathbf{W}_{\mathbb{S}})^{k,l} \; \forall \; (k,l) \neq (i,j)$ if $\mathbf{W}_{\mathbb{S}}$ is a dissimilarity matrix;
    Merge $\mathbb{S}_i[n]$ and $\mathbb{S}_j[n]$ into a single cluster,
    $\mathbb{S}[n+1] \leftarrow (\mathbb{S}[n] \backslash \{\mathbb{S}_i[n], \mathbb{S}_j[n]\}) \cup \{\mathbb{S}_i[n] \cup \mathbb{S}_j[n]\}$;
    Update $\mathbf{W}_{\mathbb{S}}$ according to the specified linkage ;
    $n \leftarrow n + 1$;
end

After having grasp the basis on agglomerative clustering, we will develop in chapter 5 an agglomerative hierarchical clustering algorithm for grouping of mobile communications basestations to be used within the scope of this project.

4.3.2 Divisive hierarchical clustering

As we said, divisive hierarchical clustering techniques start at the top of the cluster hierarchy tree. Mathematically, this means that $\mathbb{S}[0] = \{\mathbb{X}\}$. In other words, $\mathbb{S}_1[0] = \mathbb{X}$. In each iteration n, one or more of the clusters in the current partition $\mathbb{S}[n]$ are chosen and split in L pieces by using another clustering algorithm which could be K-means, Spectral Clustering, Expectation Maximization or whatever other choice we can think of.

For now, we have already seen two choices that have to be made which will have a great impact in the algorithms's performance: which clusters do we pick to be split in each iteration, and how to split them (which algorithm and in how many pieces).

For the problematic of choosing the clusters to be split, there are, as usual, a great number

of possible criteria. To name a few:

Select a cluster randomly: The simplest possible approach. The cluster to be split is chosen at random. Easy to implement but clearly suboptimal.

Split all clusters: Another very possible approach. In each iteration, we will split all of the clusters which contain more than one object until we reach the bottom of the tree where there are no more clusters with more than one point, which signals the stopping criterion.

Split the biggest cluster: With this criterion, in each iteration we look for the cluster $\mathbb{S}_i$ with the biggest cardinality, $i = \max_{j=1,\dots,k} |\mathbb{S}_j[n]|$, and split it.

Split the cluster with lowest average self-similarity: We can compute the average self-similarity of a cluster as:

$$\frac{1}{|\mathbb{S}_i[n]|^2} \sum_{j:x_j \in \mathbb{S}_i[n]} \sum_{k:x_k \in \mathbb{S}_i[n]} (\mathbf{W}_{\mathbb{S}})^{j,k} \tag{4.27}$$

where $\mathbf{W}_{\mathbb{S}}$ is the similarity matrix of the data set $\mathbb{X}$ as it was defined in the introduction. The idea is that a low average self-similarity indicates that the cluster is not homogeneous enough, so it should be split.

Other criteria can be found in [25].

On the other hand, the choice of the algorithm to split the cluster gives raise to a big family of divisive hierarchical clustering algorithms. A popular example is bisecting K-means, which is summarized in algorithm 4.3.

Algorithm 4.3: Bisecting K-means algorithm

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \mid \mathbf{x}_i \in \mathbb{R}^d$
input : Desired number of clusters K
output: A sequence of hierarchical partitions $\left\{\mathbb{S}[n] = \left\{\mathbb{S}_1[n], \mathbb{S}_2[n], \dots, \mathbb{S}[n]_{k[n]}\right\}\right\}_{n=1}^{T}$

Initialize the first partition to $\mathbb{S}[0] \leftarrow \{\mathbb{X}\}$;
$n \leftarrow 0$;
while *number of clusters in* $\mathbb{S}[n]$ *is less than* K **do**
    Select a cluster $\mathbb{S}_i[n]$ in $\mathbb{S}[n]$;
    Apply K-means to the data set $\mathbb{S}_i[n]$ with $K = 2$ to obtain two clusters $\mathbb{S}_{i,a}[n]$ and $\mathbb{S}_{i,b}[n]$ such that $\mathbb{S}_i[n] = \mathbb{S}_{i,a}[n] \cup \mathbb{S}_{i,b}[n]$ $\mathbb{S}_{i,a}[n] \cap \mathbb{S}_{i,b}[n] = \emptyset$;
    Build $\mathbb{S}[n+1] \leftarrow \left\{\mathbb{S}_1[n], \dots, \mathbb{S}_{i-1}[n], \mathbb{S}_{i,a}[n], \mathbb{S}_{i,b}[n], \mathbb{S}_{i+1}[n], \dots, \mathbb{S}[n]_{k[n]}\right\}$ $n \leftarrow n+1$;
end

A simple example of how bisecting K-means operates is shown in figure 4.14.

The criterion to choose the cluster to be split in each iteration can be any of the ones discussed before or another. One particular choice, given its relation to the objective function in

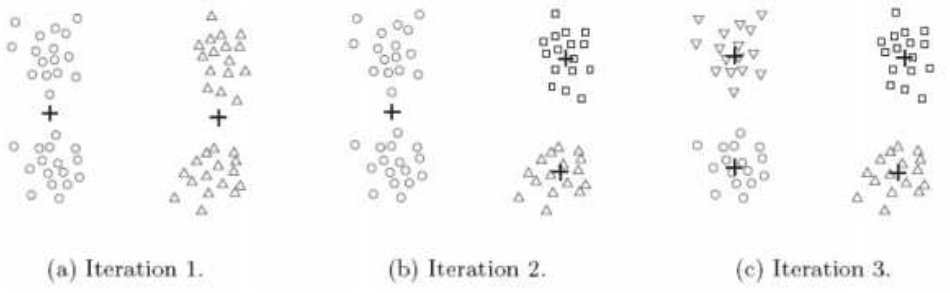


Figure 4.14: Behavior of bisecting K-means algorithm for a simple data set

K-means, is to select the cluster with the biggest sum of distances squared between the points in the cluster and the cluster's centroid. That is, in the n-th iteration we select the cluster $\mathbb{S}_i[n]$ if $C_i \geq C_j \; \forall j \neq i$ with C_i as defined in equation (4.14). Another possible modification of the algorithm, inspired on the sensitivity of K-means to initialization, is to repeat the bisection process at each iteration M times and keep the bisection which yields a lowest value for the K-means cost function (4.14).

Most of the algorithms actually bisect the clusters to be split, that is, they choose $L = 2$. Also, it is not necessary to generate the whole hierarchical tree. Just like bisecting K-means, we can choose to stop when a certain prefixed number of clusters K is generated, or, as we will do in one of the novel algorithms developed for this project, stop when the cardinality of all the clusters is below a prefixed threshold.

Apart from bisecting K-means, there are many other divisive hierarchical clustering algorithms. In chapter 5, we will propose a novel Spectral Clustering based divisive hierarchical algorithm for automatic adaptive grouping of mobile communications base-stations for MIMO coordination.

4.4 Spectral clustering

Spectral clustering represents a family of clustering algorithms which are considered the current state-of-the-art in clustering techniques. As we will discuss later in this section, spectral clustering has a number of advantages with respect to the traditional clustering algorithms like K-means variants or agglomerative hierarchical clustering. Spectral clustering is both really simple to implement and understand, requiring only some basic notions of graph theory and linear algebra, and usually outperforms the traditional clustering methods in practically all scenarios.

During the remainder of this section, we will first introduce the basic concepts of graph theory required to understand spectral clustering, to then derive spectral clustering from scratch with the different variants we can find, showing a summary of the most well-known spectral clustering algorithms in the literature. We will conclude this section with a brief introduction to a fascinating topic: all the connections between spectral clustering and other machine learning concepts, which give a really valuable insight on the reason why this family of algorithms perform

so well.

4.4.1 Graph theory

4.4.1.1 Basic concepts and notation

In graph theory, a graph $G = (V, E)$ is a set of N vertexes $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ together with the set of edges between the vertexes, E. Some basic concepts and definitions about graphs are:

Weighted/Unweighted graphs: When each edge from a vertex $\mathbf{v}_i$ to a vertex $\mathbf{v}_j$ carries a non-negative weight w_{ij}, we say that the graph G is a weighted graph. If there is no edge from vertex $\mathbf{v}_i$ to vertex $\mathbf{v}_j$, then $w_{ij} = 0$. On the other hand, unweighted graphs are simpler structures where an edge either exists or not. We can treat them as weighted graphs for which the weight of each potential edge from a vertex $\mathbf{v}_i$ to a vertex $\mathbf{v}_j$ takes a binary value: $w_{ij} = 1$ if the edge exists and $w_{ij} = 0$ if the edge does not exist.

(Weighted) adjacency matrix: We define the (weighted) adjacency matrix as a N-by-N matrix $\mathbf{W}$ such that $(\mathbf{W})^{i,j} = w_{ij}$. If the graph is unweighted, then $\mathbf{W}$ is a binary matrix, that is, its entries are either 1 or 0.

Directed/Undirected graphs: When a graph is undirected, edges are bidirectional. This means that if an edge from vertex $\mathbf{v}_i$ to vertex $\mathbf{v}_j$ exists, another edge (with the same weight if the graph is weighted) must exist from vertex $\mathbf{v}_j$ to vertex $\mathbf{v}_i$. We can then refer to both as edges as the edge between $\mathbf{v}_i$ and $\mathbf{v}_j$ without ambiguity. However, directed graphs are more complex structures that lack this symmetry.

Note that, for a directed graph, the adjacency matrix is a symmetric matrix, that is, $\mathbf{W} = \mathbf{W}^T$.

Degree of a vertex: The degree of a vertex $\mathbf{v}_i \in V$ is defined as:

$$d_i = \sum_{j=1}^{N} w_{ij} \tag{4.28}$$

Note that we can compactly compute an N-dimensional column vector with the degrees of all points, $\mathbf{d} = [d_1, d_2, \ldots, d_N]^T$ in the data set as:

$$\mathbf{d} = \mathbf{W}\mathbf{1} \tag{4.29}$$

Where $\mathbf{1}$ is the all-ones vector of the appropriate size.

Since $w_{ij} \neq 0$ if and only if there is an edge from vertex $\mathbf{v}_i$ to vertex $\mathbf{v}_j$, the degree of a vertex measures the sum-weight of all the edges originated in vertex $\mathbf{v}_i$.

Indicator vector: For a subset $V_i \subset V$, we define the indicator vector of V_i as a N-dimensional vector $\mathbf{e}_i = \left[\mathbf{e}_i^1, \mathbf{e}_i^2, \ldots, \mathbf{e}_i^N\right]^T$ such that $\mathbf{e}_i^j = 1$ if and only if vertex $\mathbf{v}_j$ belongs to subset V_i and $\mathbf{e}_i^j = 0$ otherwise. Note that the indicator vector of V is simple the all ones vector $\mathbf{1}$. Intuitively, the less connected a vertex $\mathbf{v}_u$ is, the smaller its degree.

Cardinality of a set of vertexes: The cardinality of a subset $V_i \subset V$ will be denoted by $|V_i|$ and, since V_i is a finite set, it equals the number of vertexes in the subset.

Volume of a set of vertexes: The volume of a subset $V_i \subset V$ will be denoted by $\mathrm{vol}\,(V_i)$ and is defined to be the sum of the degrees of all vertexes in the subset, that is:

$$\mathrm{vol}\,(V_i) = \sum_{\{j|\mathbf{v}_j \in V_i\}} d_j \tag{4.30}$$

Connected subset: A subset $V_i \subset V$ is said to be connected if any vertex $\mathbf{v}_j$ in V_i can be reached from any other vertex $\mathbf{v}_k$ in V_i by traveling a path of edges between vertexes of V_i. In other words, in a connected subset, we can move freely from one vertex to another following a path of edges without getting outside of the subset at any moment. A subset is called unconnected whenever it is not connected.

Connected component: A subset $V_i \subset V$ is called a connected component of a graph when it is connected and, moreover, there are no edges between any point of V_i and points in the complement of V_i, $\overline{V_i}$ where $\overline{V_i} = \{\mathbf{v}_j \in V : \mathbf{v}_j \notin V_i\}$. In other words, there is no way to travel from points inside V_i to points outside the subset.

Hard partition of a graph: A hard partition of a graph is a hard partition of the vertexes in the same sense we discussed in the introduction of this chapter. That is, a collection of subsets $\mathbb{V} = \{V_1, V_2, \ldots, V_k\}$ such that $V_i \cap V_j = \emptyset \; \forall \; i \neq j$ and $V_1 \cup V_2 \cup \ldots \cup V_k = V$.

4.4.1.2 Graph Laplacians

The graph Laplacian matrix is a fundamental concept in graph theory and will also be an essential tool to derive spectral clustering. In the literature, there are several different definitions of the graph Laplacian matrix which are perfectly valid as long as the subsequent derivations are consistent with the initial definition. Even if they are different, they are very similar both in definition and properties. In this work, we will follow the convention in [2] and [3] defining the graph Laplacian matrix of an undirected weighted graph G as:

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{4.31}$$

Where $\mathbf{D} = \mathrm{diag}\,(\mathbf{d})$ and $\mathbf{W}$ the weighted adjacency matrix of the graph. A detailed discussion on the properties of $\mathbf{L}$ can be found in [26]. However, we will stick to a summary of the properties which are relevant for spectral clustering, taken from [2]:

Theorem 1.

(i) $\mathbf{x}^T\mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(x_i - x_j)^2 \; \forall \; \mathbf{x} \in \mathbb{R}^N$.

(ii) $\mathbf{L}$ *is a positive semi-definite, symmetric matrix.*

(iii) 0 *is always an eigenvalue of* $\mathbf{L}$ *and its corresponding eigenvector is* $\mathbf{1}$.

(iv) The algebraic multiplicity of the eigenvalue 0 *of* $\mathbf{L}$ *equals the number of connected components in the associated graph* G*. Moreover, the associated eigenvectors are the indicator vectors of each of the connected components.*

Proof.

(i) From the definition of $\mathbf{L}$, $\mathbf{x}^T\mathbf{L}\mathbf{x} = \mathbf{x}^T\mathbf{D}\mathbf{x} - \mathbf{x}^T\mathbf{W}\mathbf{x}$. Since $\mathbf{D}$ is a diagonal matrix, we can express both quadratic forms as:

$$\begin{aligned}\mathbf{x}^T\mathbf{D}\mathbf{x} - \mathbf{x}^T\mathbf{W}\mathbf{x} &= \sum_{i=1}^{N} d_i x_i^2 - \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}x_i x_j = \\ &= \frac{1}{2}\left(\sum_{i=1}^{N} d_i x_i^2 - 2\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}x_i x_j + \sum_{j=1}^{N} d_j x_j^2\right) = \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(x_i - x_j)^2\end{aligned}$$

(ii) Since $\mathbf{L} = \mathbf{D} - \mathbf{W}$ with $\mathbf{D}$ diagonal and $\mathbf{W}$ symmetric for undirected graphs, then $\mathbf{L}$ is symmetric too. From (i), we see that $\mathbf{x}^T\mathbf{L}\mathbf{x} \geq 0 \; \forall \; \mathbf{x} \in \mathbb{R}^N$

(iii) We have that $\mathbf{L}\mathbf{1} = \mathbf{D}\mathbf{1} - \mathbf{W}\mathbf{1} = \mathbf{d} - \mathbf{d} = \mathbf{0}$. Therefore, $\mathbf{1}$ belongs to the null space of $\mathbf{L}$, making it an eigenvector of $\mathbf{L}$ with eigenvalue 0.

(iv) Let us assume that $\mathbf{x}^T\mathbf{L}\mathbf{x} = 0$. Then, by (i), this means that:

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(x_i - x_j)^2 = 0$$

In particular, since $w_{ij} \geq 0 \; \forall \; i, j$, we must have that $w_{ij}(x_i - x_j)^2 = 0 \; \forall \; i = 1, \ldots, N \; j = 1, \ldots, N$. This is completely equivalent to saying that $x_i = x_j$ if $w_{ij} > 0$. In words, if vertexes i and j are connected by an edge in the graph, $\mathbf{x}$ will not belong to the null space of $\mathbf{L}$ unless its entries x_i and x_j are equal. Moreover, if j is also connected to another vertex k, by the same reasoning $x_j = x_k$, so that we end up with $x_i = x_j = x_k$.

Extending this argument, if an eigenvector of $\mathbf{L}$ with eigenvalue 0, $\mathbf{x}$ has non-zero entry x_i, then all the entries in $\mathbf{x}$ corresponding to vertexes in the same connected component as vertex i must have the same value as x_i.

With this procedure, if a graph has k connected components, we can construct up to k linearly independent vectors in the null space of $\mathbf{L}$, being the set of indicator vectors of the connected components a basis of such null space.

□

It is usual in the literature to normalize the graph Laplacian matrix. Different normalizations will lead to variations of spectral clustering algorithms as we shall see later. Some typical ways to build normalized Laplacians are:

$$\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} \tag{4.32}$$

And:

$$\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W} \tag{4.33}$$

The notations are due to the fact that $\mathbf{L}_{sym}$ is still a symmetric matrix, and $\mathbf{L}_{rw}$ is related to a random walk interpretation which we will discuss later. The properties of both matrices are very similar to those of the unnormalized graph Laplacian $\mathbf{L}$. It is obvious from the definition of the normalized graph Laplacians that $\mathbf{L}_{rw}$ shares the null space of $\mathbf{L}$ and that $\mathbf{L}_{sym}$ also behaves very much alike, with the particularity that if $\mathbf{x}$ is in the null space of $\mathbf{L}$ and $\mathbf{L}_{rw}$, then the corresponding the corresponding null-space vector of $\mathbf{L}_{sym}$ would be $\mathbf{D}^{\frac{1}{2}}\mathbf{x}$. In particular, this means that the multiplicity of eigenvalue 0 of both normalized matrices keeps its meaning. The reader can find a more detailed discussion on the exact relation between the spectrums of $\mathbf{L}$, $\mathbf{L}_{sym}$ and $\mathbf{L}_{rw}$ with the corresponding proofs in one of the two main sources in which most of this chapter is based, [2].

4.4.2 Interplay between graph theory and spectral clustering

As the reader may have already realized, a graph provides essentially the same representation of a data set as the similarity matrix of a data set as defined at the beginning of this chapter. Indeed, if we consider that each object $\mathbf{x}_i$ in the data set is a vertex $\mathbf{v}_i$ in the graph, we can encode the information about the similarity between objects in the data set in the set of edges. A summary of the most typical ways to build a graph from a data set are:

Fully connected graph: We can build an undirected weighted graph where each vertex $\mathbf{v}_i$ is connected by an edge to all other vertexes $\mathbf{v}_j$ in such a way that the weight of the edge between two vertexes $\mathbf{v}_i$ and $\mathbf{v}_j$ equals the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. In this case, the weighted adjacency matrix of the graph and the similarity matrix of the data set are identical. The granularity of this kind of graph is controlled via the similarity function.

ϵ-neighborhood graph: Alternatively, it is possible to obtain a more abstract representation of the data set by using an undirected unweighted graph where two vertexes $\mathbf{v}_i$,$\mathbf{v}_j$ are

connected by an edge if and only if the dissimilarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ is below a prefixed threshold level ϵ (alternatively, we can connect two vertexes when the similarity is above a certain threshold if we don't use dissimilarities). In this kind of scheme, the information about the proximity of points is encoded in the existence of edges, so that's why it is usually built as an unweighted graph. However, we could still weight the edges by the similarities if we wanted to, making the graph weighted.

The level of detail is controlled by the parameter ϵ in such a way that a small value for ϵ will result in a graph with too many connected components, whereas a value which is too big may link dissimilar groups and even outliers. As with all machine learning algorithms, we should experiment with several choices and cross-validate to find the optimal value for the parameter.

k-nearest neighbor graph: We build an undirected graph by connecting two vertexes $\mathbf{v}_i$,$\mathbf{v}_j$ if $\mathbf{x}_i$ is among the k-nearest neighbors of $\mathbf{x}_j$ OR $\mathbf{x}_j$ is among the k-nearest neighbors of $\mathbf{x}_i$. We can weight the resulting edges between vertexes with the similarity between the corresponding data points or not. In this case, k will control the granularity of the data representation in a similar manner as ϵ in the ϵ-neighborhood graph. A typical rule of thumb is to employ $\epsilon \approx \log(N)$, but again, the best idea is to try several values and pick the one which behaves better for our application, [2].

Mutual k-nearest neighbor graph: We build an undirected graph by connecting two vertexes $\mathbf{v}_i$,$\mathbf{v}_j$ if $\mathbf{x}_i$ is among the k-nearest neighbors of $\mathbf{x}_j$ AND $\mathbf{x}_j$ is among the k-nearest neighbors of $\mathbf{x}_i$. Therefore, the mutual k-nearest neighbor graph imposes more restrictive conditions to place an edge between two vertexes than the k-nearest neighbor graph. We also have the freedom to make the graph weighted or unweighted, so that we can choose whatever fits better the needs of our particular application. For this kind of approach, there are no popular rules of thumb for the choice of k. However, as an initial setup, we should use a value for k significantly larger than the equivalent for the k-nearest neighbor graph, since the mutual k-nearest neighbor graph usually has much fewer edges.

An example of how different graphs can be built from the same data set by using some of the conventions we just discussed can be seen in figure 4.15.

Intuitively, the fully connected graph preserves all the information, whereas the other schemes "filter" the data set by eliminating similarities between points which are too far apart. This approach is beneficial in the presence of noise and outliers, since they would not affect the graph shape. On the other hand, we may be pruning relevant information and ending up with a poorer representation of the original data. Another consideration is that both the ϵ-neighborhood graph and the k-nearest neighbor based algorithms result in weighted adjacency matrices which are sparse, with the subsequent computational advantages. However, there is no general answer about which kind of graph performs better nor theoretical results about how the performance

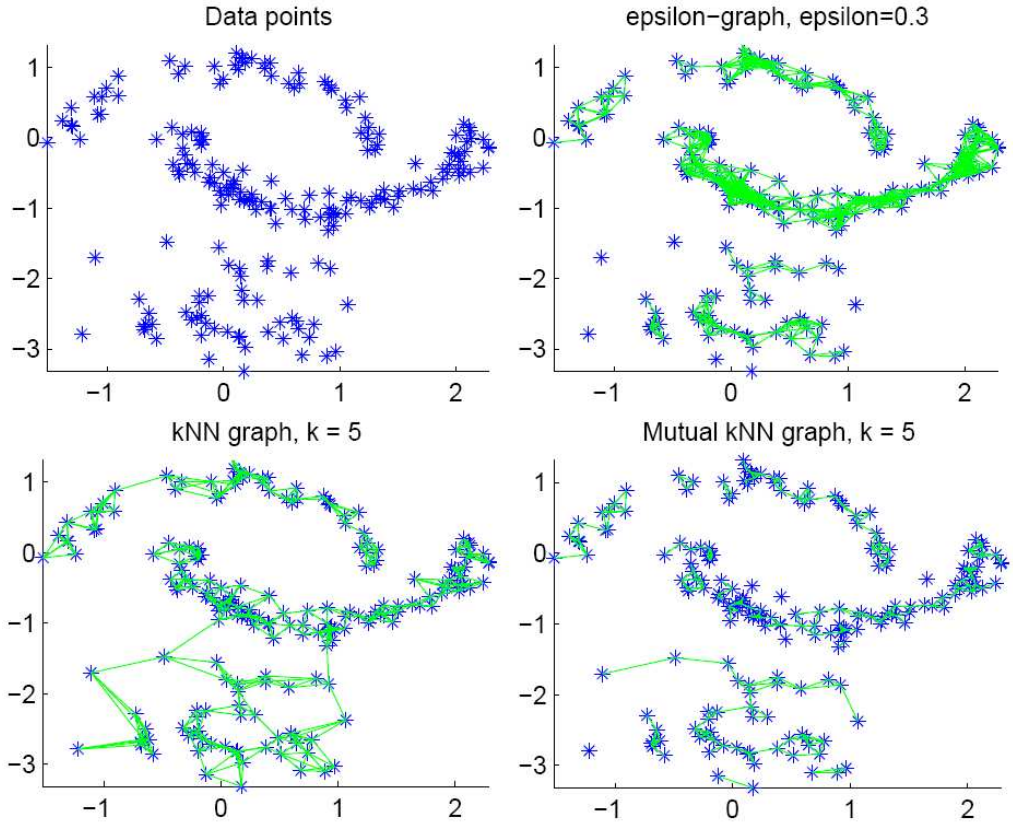


Figure 4.15: Different graphs obtained from the same dataset.

changes with the type of graph. Therefore, the engineer shall try several options and cross-validate the results to make a choice.

We may ask ourselves why we should bother to change the representation of the data set if we are going to end up with the same thing. The answer lies in the fact that, by using a graph representation, we may formulate clustering as the optimization of a family of cost functions treated in classical graph theory, the so called graph cuts.

Graph cuts

From an intuitive point of view, to cluster data we want to find a partition of the data set so that points belonging to the same cluster are very similar to each other and points belonging to different cluster have a low similarity. If we restate that objective in terms of graph theory, what we want is to find a partition so that the edges between points in the same subset have large weights but the edges between points in different subsets have small weights. In the ideal case, following the terminology exposed before, we would like each cluster to be a connected component of the graph. In most cases, a partition like that does not exist. However, we can easily build cost functions to quantify the concept of maximizing the sum of intra-cluster weights and minimizing the sum of inter-cluster weights and try to optimize them.

First of all, to simplify the notation, we will introduce:

$$W(A,B) = \frac{1}{2} \sum_{\{i|\mathbf{v}_i \in A\}} \sum_{\{j|\mathbf{v}_j \in B\}} w_{ij} \tag{4.34}$$

With that definition, $W(A,B)$ measures the sum of the weights of all edges between vertexex in subset A to vertexex in subset B. Since the graph is undirected, the factor $\frac{1}{2}$ avoids counting each edge twice (one per each direction). However, since it will become merely a scaling of the cost function, we could perfectly drop it without changing the results.

Note that the definition still holds if sets A and B overlap and even if $A = B$. Precisely, one of the particular cases of interest is the value of $W(A,A)$, which accounts for the sum of the weights of all edges between any two vertexes in subset A, that is, for clustering, we seek to maximize $W(V_i,V_i)$ for all the V_i in the partition $\mathbb{V}$, so that points in the same cluster are similar. The other case of interest is $W\left(A,\overline{A}\right)$, which accounts for the sum of the weights of all edges between points in subset A and points outside A. A good partition for cluster should then achieve low values of $W\left(V_i,\overline{V_i}\right)$ for all the V_i in the partition $\mathbb{V}$, meaning that data in different clusters are dissimilar.

Using this notation, we build the following cost functions:

MinCut: The Cut objective function is simply the sum of the weights of all edges between points assigned to different clusters. Mathematically:

$$\text{cut}(\mathbb{V}) = \sum_{i=1}^{k} W\left(V_i,\bar{V}_i\right) \tag{4.35}$$

We can easily see that minimizing the value of $\text{cut}(\mathbb{V})$ is equivalent to finding a partition such as the generated clusters are as dissimilar as possible. That is why this method is called MinCut.

A priori, MinCut seems to be a reasonable approach to the problem. Moreover, for the particular case in which $k = 2$, it can be solved efficiently ([27]). Nevertheless, if we think about it more throughly, we can easily construct sensible scenarios where this cost function is extremely pathological. The exact problem is that, much more frequently than we would like, the optimal solution to the MinCut problem simply separates one point from the rest of the graph. This led to the idea of trying to include some kind of penalization for partitions with unbalanced clusters (clusters are said to be unbalanced whenever they differ greatly in size).

RatioCut: The RatioCut objective function tries to tackle the problem discussed before by using the cardinality of each cluster as a measure of size. The objective function becomes:

$$\text{RatioCut}(\mathbb{V}) = \sum_{i=1}^{k} \frac{W\left(V_i,\overline{V_i}\right)}{|V_i|} \tag{4.36}$$

As we see, the sum-weigh of the edges going outside a cluster V_i is normalized by the cardinality of the subset, hence the name of RatioCut. In this way, when a cluster V_i is very small, the normalized sum-weight $\frac{W\left(V_i,\overline{V_i}\right)}{|V_i|}$ will be big even if the original sum-weight of $W\left(V_i,\overline{V_i}\right)$ was small.

Moreover, since the function $\sum_{i=1}^{k}\frac{1}{x_i}$ is minimized when $x_1 = x_2 = \ldots = x_k$, we see that RatioCut somehow benefits balanced partitions.

NormalizedCut: Normalized cut was first introduced in [4]. The idea is very similar to that of RatioCut, but instead of normalizing $W\left(V_i,\overline{V_i}\right)$ by the cardinality of V_i, we normalize it by its volume:

$$\mathrm{NCut}\left(\mathbb{V}\right) = \sum_{i=1}^{k}\frac{W\left(V_i,\overline{V_i}\right)}{\mathrm{vol}\left(V_i\right)} \tag{4.37}$$

Normalized cut also gives a partition with balanced clusters. However, whereas in RatioCut the clusters were balanced in the sense of containing a similar number of points, NormalizedCut balances clusters by trying to find a partition so that the sum of the weights of all edges in each cluster is similar for all the clusters.

PenalizedCut: In [3] the authors propose a generalization of both RatioCut and NormalizedCut which they call PenalizedCut. The idea is very simple. In RatioCut, whenever a point $\mathbf{v}_i$ is added to a certain cluster, the normalization factor for that cluster increases by 1. On the other hand, in NCut, it would increase by the degree of that point, d_i. The proposition of that paper is to allow a custom normalization by defining a set of values $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_N]^T$, in such a way that adding the point to a cluster increases the normalization by π_i begin the only restriction the positivity of the weights, $\pi_i \geq 0$. The normalization factor for a cluster V_i can then be expressed in a compact way using the indicator vectors of the cluster as $\boldsymbol{\pi}^T\mathbf{e}_i$. Taking that into account, PenalizedCut becomes:

$$\mathrm{PCut}\left(\mathbb{V}\right) = \sum_{i=1}^{k}\frac{W\left(V_i,\overline{V_i}\right)}{\boldsymbol{\pi}^T\mathbf{e}_i} \tag{4.38}$$

PenalizedCut has the fundamental advantage of being a generalization since, if we let $\boldsymbol{\pi} = \mathbf{1}$ we recover RatioCut and defining $\boldsymbol{\pi} = \mathbf{d}$ we get NormalizedCut.

Sadly, the only graph cut which is easy to solve is precisely the one which is not useful for clustering, MinCut. PenalizedCut, including the two particular cases of RatioCut and NormalizedCut, are NP hard problems as it was shown in [28]. However, as we shall see in the next section, spectral clustering uses a clever trick to approximate the optimal solution of graph cut problems using a spectral representation (hence the name) of the associated graph Laplacian matrix.

4.4.3 Approximating graph cuts with linear algebra: spectral relaxation

As we have seen, graph cuts are very representative objective functions for clustering. However, they are NP-hard problems, so that they are usually unfeasible for real-life scenarios. In the recent years, spectral clustering has become really popular since it provides a feasible way of approximating the solution of graph cut objective functions.

First, a matrix representation of the objective functions will be obtained. This will allow us to cast it into a form which resembles typical problems in linear algebra whose solution is well-known. This new form will obviously be still NP-hard but it will clearly suggest a way of approximating the solution. But, we will see that, by dropping one particular condition, we obtain a very typical constrained optimization problem with matrix variables, whose solution will be strongly linked to the spectrum of the graph Laplacian matrix. To sum up, we *relax* the problem into a tractable form, whose solution can be interpreted in terms of the spectrum of a matrix. That's why this step is usually denoted by *spectral relaxation*. Sadly, the condition we will get rid of is precisely the one which ensures that the solution actually represents a partition of the data set. As we will see, we can prove in a lot of different ways (this is precisely one of the most fascinating aspects of spectral clustering!) that the relaxed solution makes sense and is interpretable in terms of clustering, only just not in a straightforward way. The step in which extract information to encode a partition from the solution of the relaxed problem is usually denoted *rounding*, since it will involve a kind of discretization of the solution.

We will begin by finding a matrix representation of the graph cuts problems. Without loss of generality, we will focus on PenalizedCut, since RatioCut and NormalizedCut are particular cases of PenalizedCut.

In order to find the sought matrix representation of PCut, we will define $\mathbf{\Pi} = \operatorname{diag}(\boldsymbol{\pi})$, a diagonal matrix whose diagonal entries are the customized per-point normalization factors π_i and, more importantly, the N-by-k cluster assignation matrix $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k]$. Note that $(\mathbf{E})^{i,j}$ will have value 1 if point i belongs to the j-th cluster and 0 otherwise. Since we only consider partitional clustering in this section, $\mathbf{E}$ is then a binary matrix with the extra restriction that it may have only one entry with value 1 in each row.

Taking into account this two definitions and using property (i) in theorem 1, we have that:

$$W\left(V_i, \overline{V_i}\right) = \mathbf{e}_i^T \mathbf{L} \mathbf{e}_i = \left(\mathbf{E}^T \mathbf{L} \mathbf{E}\right)^{i,i} \tag{4.39}$$

On the other hand, the normalization factor in PenalizedCut can also be written as:

$$\boldsymbol{\pi}^T \mathbf{e}_i = \mathbf{e}_i^T \mathbf{\Pi} \mathbf{e}_i = \left(\mathbf{E}^T \mathbf{\Pi} \mathbf{E}\right)^{i,i} \tag{4.40}$$

Putting both things together and defining PenalizedCut can be expressed as:

$$\operatorname{PCut}(\mathbb{V}) = \sum_{i=1}^{k} \frac{W\left(V_i, \overline{V_i}\right)}{\boldsymbol{\pi}^T \mathbf{e}_i} = \sum_{i=1}^{k} \frac{\left(\mathbf{E}^T \mathbf{L} \mathbf{E}\right)^{i,i}}{\left(\mathbf{E}^T \mathbf{\Pi} \mathbf{E}\right)^{i,i}} = \operatorname{Tr}\left(\mathbf{E}^T \mathbf{L} \mathbf{E} \left(\mathbf{E}^T \mathbf{\Pi} \mathbf{E}\right)^{-1}\right) \tag{4.41}$$

The latter equality follows from the fact that, with the definitions above, $\mathbf{E}^T\mathbf{\Pi}\mathbf{E}$ is a diagonal matrix.

At this point, the objective function of PenalizedCut is already expressed in terms of linear algebra. However, the current result is still not satisfactory since we would like to express it as a problem with a well-known solution. We can cast equation (4.41) into something easier to handle using the following theorems.

Theorem 2. *The objective function* $\mathrm{PCut}(\mathbb{V}) = \mathrm{Tr}\left(\mathbf{E}^T\mathbf{L}\mathbf{E}\left(\mathbf{E}^T\mathbf{\Pi}\mathbf{E}\right)^{-1}\right)$ *can be expressed as* $\mathrm{PCut}(\mathbb{V}) = \mathrm{Tr}\left(\mathbf{H}^T\mathbf{L}\mathbf{H}\right)$ *for any matrix* $\mathbf{H} \in \mathbb{R}^{N\times k}$ *satisfying:*

(a) The columns of $\mathbf{H}$ *are piecewise constant with respect to the indicator vectors of* $\mathbb{V}$ *in* $\mathbf{E}$*. That is,* $\mathbf{H} = \mathbf{E}\mathbf{\Psi}$ *for any non-singular* $\mathbf{\Psi} \in \mathbb{R}^{k\times k}$*.*

(b) $\mathbf{H}^T\mathbf{\Pi}\mathbf{H} = \mathbf{I}_k$*.*

Proof. On the one hand, using condition (a) of $\mathbf{H}$ we have that:

$$\mathrm{Tr}\left(\mathbf{H}^T\mathbf{L}\mathbf{H}\right) = \mathrm{Tr}\left(\mathbf{\Psi}^T\mathbf{E}^T\mathbf{L}\mathbf{E}\mathbf{\Psi}\right) = \mathrm{Tr}\left(\mathbf{E}^T\mathbf{L}\mathbf{E}\mathbf{\Psi}\mathbf{\Psi}^T\right)$$

Also, from condition (b) we can see:

$$\mathbf{I}_k = \mathbf{H}^T\mathbf{\Pi}\mathbf{H} = \mathbf{\Psi}^T\mathbf{E}^T\mathbf{\Pi}\mathbf{E}\mathbf{\Psi} \implies \mathbf{E}^T\mathbf{\Pi}\mathbf{E} = \left(\mathbf{\Psi}\mathbf{\Psi}^T\right)^{-1} \iff \mathbf{\Psi}\mathbf{\Psi}^T = \left(\mathbf{E}^T\mathbf{\Pi}\mathbf{E}\right)^{-1}$$

Therefore, putting both arguments together we can conclude that:

$$\mathrm{PCut}(\mathbb{V}) = \mathrm{Tr}\left(\mathbf{E}^T\mathbf{L}\mathbf{E}\left(\mathbf{E}^T\mathbf{\Pi}\mathbf{E}\right)^{-1}\right) = \mathrm{Tr}\left(\mathbf{H}^T\mathbf{L}\mathbf{H}\right)$$

Which concludes the proof. □

The reader may wonder whether a matrix $\mathbf{H}$ satisfying the hypothesis of theorem 2 exists. However, from the proof of the theorem, we can see that this amounts to showing if we can find $\mathbf{\Psi} \in \mathbb{R}^{k\times k}$ such that $\mathbf{\Psi}\mathbf{\Psi}^T = \left(\mathbf{E}^T\mathbf{\Pi}\mathbf{E}\right)^{-1}$. Since $\left(\mathbf{E}^T\mathbf{\Pi}\mathbf{E}\right)^{-1}$ is a diagonal matrix, which we may call $\mathbf{B}$, this is equivalent to asking that $\mathbf{\Psi}\mathbf{\Psi}^T = \mathbf{B}$. It is then straightforward to see that any matrix of the form $\mathbf{\Psi} = \mathbf{B}^{1/2}\mathbf{U}$ with $\mathbf{U}$ an arbitrary $k \times k$ orthonormal matrix, satisfies the requirements.

At this point, we have successfully found the sought matrix-representation of PenalizedCut, which may be finally expressed as:

$$\min_{\mathbf{H}|\mathbf{H}=\mathbf{E}\mathbf{\Psi}} \mathrm{Tr}\left(\mathbf{H}^T\mathbf{\Lambda}\mathbf{H}\right) \quad s.t. \quad \mathbf{H}^T\mathbf{\Pi}\mathbf{H} = \mathbf{I}_k \tag{4.42}$$

At first sight, the previous optimization problem looks fantastic: it seems a simple problem in which the trace of a quadratic form needs to be minimized over a manifold. However, if we look closely at the domain of the problem, $\mathbf{H}|\mathbf{H} = \mathbf{E}\mathbf{\Psi}$, we see that things are much more tricky. The rigorous definition of the problem only considers matrices $\mathbf{H}$ which are piecewise

constant on $\mathbf{E}$, where $\mathbf{E}$ must be a valid cluster-assignation matrix as defined at the beginning of this section ($(\mathbf{E})^{i,j} = 1$ if point i belongs to the j-th cluster and $(\mathbf{E})^{i,j} = 0$ otherwise). Solving this problem is still NP-hard, so it would seem that we have gained nothing by all this work. Nevertheless, even if it may seem surprising, simply by finding the solution of the problem over $\mathbf{H} \in \mathbb{R}^{N\times k}$ we obtain a result which contains a lot of information about the sought partition.

Therefore, following that idea, we apply *spectral relaxation* to obtain the following *relaxed* problem:

$$\min_{\mathbf{H}\in\mathbb{R}^{N\times k}} \operatorname{Tr}\left(\mathbf{H}^T\mathbf{\Lambda}\mathbf{H}\right) \quad s.t. \quad \mathbf{H}^T\mathbf{\Pi}\mathbf{H} = \mathbf{I}_k \tag{4.43}$$

The solution of (4.43) is in the following theorem:

Theorem 3. *The solution of the optimization problem:*

$$\min_{\mathbf{H}\in\mathbb{R}^{N\times k}} \operatorname{Tr}\left(\mathbf{H}^T\mathbf{\Lambda}\mathbf{H}\right) \quad s.t. \quad \mathbf{H}^T\mathbf{\Pi}\mathbf{H} = \mathbf{I}_k$$

If $\mathbf{L}$ *is a real-valued symmetric matrix fulfilling* $\mathbf{\Lambda}\mathbf{1} = \mathbf{0}$ *and* $\mathbf{\Pi} = \operatorname{diag}(\boldsymbol{\pi})$ *with* $\pi_i \geq 0$ *is given by* $\mathbf{H} = \mathbf{\Pi}^{-1/2}\mathbf{U}\mathbf{Q}^T$ *where* $\mathbf{U} = [\mathbf{u}_1\mathbf{u}_2\dots\mathbf{u}_k] \in \mathbb{R}^{N\times k}$ *is a matrix whose columns are the* k *eigenvectors of the normalized graph Laplacian matrix* $\widetilde{\mathbf{L}} = \mathbf{\Pi}^{-1/2}\mathbf{L}\mathbf{\Pi}^{-1/2}$ *associated to the* k *smallest eigenvalues and* $\mathbf{Q} \in \mathbb{R}^{k\times k}$ *is an arbitrary real-valued orthonormal matrix.*

Proof. We have to solve a constrained optimization problem. Hence, we must write its associated Lagrangian:

$$\mathcal{L}(\mathbf{H}, \mathbf{\Phi}) = \operatorname{Tr}\left(\mathbf{H}^T\mathbf{\Lambda}\mathbf{H}\right) - \operatorname{Tr}\left(\mathbf{\Phi}\left(\mathbf{H}^T\mathbf{\Pi}\mathbf{H} - \mathbf{I}_k\right)\right)$$

Where we have introduced:

$$\mathbf{\Phi} = \begin{pmatrix} \phi_{1,1} & \phi_{2,1} & \cdots & \phi_{k,1} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{k,2} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{k,1} & \phi_{k,2} & \cdots & \phi_{k,k} \end{pmatrix}$$

A symmetric $k \times k$ matrix of Lagrange multipliers.

In order to simplify the expression of the Lagrangian, it will be useful to define $\widetilde{\mathbf{H}} = \mathbf{\Pi}^{1/2}\mathbf{H}$, so that we can write:

$$\mathcal{L}\left(\widetilde{\mathbf{H}}, \mathbf{\Phi}\right) = \operatorname{Tr}\left(\widetilde{\mathbf{H}}^T\widetilde{\mathbf{L}}\widetilde{\mathbf{H}}\right) - \operatorname{Tr}\left(\mathbf{\Phi}\left(\widetilde{\mathbf{H}}^T\widetilde{\mathbf{H}} - \mathbf{I}_k\right)\right)$$

With $\widetilde{\mathbf{L}} = \mathbf{\Pi}^{-1/2}\mathbf{L}\mathbf{\Pi}^{-1/2}$ being the (generalized) normalized Laplacian matrix associated to the graph.

We can know differentiate the Lagrangian with respect to $\widetilde{\mathbf{H}}$ to find:

$$\frac{\partial\mathcal{L}}{\partial\widetilde{\mathbf{H}}} = 2\widetilde{\mathbf{L}}\widetilde{\mathbf{H}} - 2\widetilde{\mathbf{H}}\mathbf{\Phi}$$

Since we try to find a stationary point of the Lagrangian, we end up with the condition:

$$\widetilde{\mathbf{L}}\widetilde{\mathbf{H}} = \widetilde{\mathbf{H}}\mathbf{\Phi}$$

As $\mathbf{\Phi}$ is a symmetric matrix, it satisfies the hypothesis of the spectral theorem for normal matrices, so we can introduce in the previous equation its unitary diagonalization $\mathbf{\Phi} = \mathbf{Q}\mathbf{S}_{\mathbf{\Phi}}\mathbf{Q}^T$ to find:

$$\widetilde{\mathbf{L}}\widetilde{\mathbf{H}}\mathbf{Q} = \widetilde{\mathbf{H}}\mathbf{Q}\mathbf{S}_{\mathbf{\Phi}}$$

But the previous equation is simply stating that the columns of $\widetilde{\mathbf{H}}\mathbf{Q}$ are k of the eigenvectors of $\widetilde{\mathbf{L}}$ with corresponding eigenvalues given by the diagonal entries of the $k \times k$ diagonal matrix $\mathbf{S}_{\mathbf{\Phi}}$. In other words, the eigenvalues of $\mathbf{\Phi}$ coincide with those of $\widetilde{\mathbf{L}}$. If we introduce $\mathbf{U} = [\mathbf{u}_1\mathbf{u}_2 \dots \mathbf{u}_k] \in \mathbb{R}^{N\times k}$, such that $\mathbf{U} = \widetilde{\mathbf{H}}\mathbf{Q}$, that is, a matrix whose k columns are k of the eigenvectors of $\widetilde{\mathbf{L}}$, we can solve for $\widetilde{\mathbf{H}}$ to find that $\widetilde{\mathbf{H}} = \mathbf{U}\mathbf{Q}^T$. Moreover, it is straightforward to check that with that solution:

(a) The constraint is readily satisfied: $\widetilde{\mathbf{H}}^T\widetilde{\mathbf{H}} = \mathbf{Q}\mathbf{U}^T\mathbf{U}\mathbf{Q}^T = \mathbf{I}_k$ since both $\mathbf{U}$ and $\mathbf{Q}$ have orthonormal columns.

(b) The value of the objective function becomes:

$$\mathrm{Tr}\left(\mathbf{H}^T\mathbf{\Lambda}\mathbf{H}\right) = \mathrm{Tr}\left(\widetilde{\mathbf{H}}^T\widetilde{\mathbf{L}}\widetilde{\mathbf{H}}\right) = \mathrm{Tr}\left(\mathbf{Q}\mathbf{U}^T\widetilde{\mathbf{L}}\mathbf{U}\mathbf{Q}^T\right) = \mathrm{Tr}\left(\mathbf{Q}\mathbf{U}^T\mathbf{U}_{\widetilde{\mathbf{L}}}\mathbf{S}_{\widetilde{\mathbf{L}}}\mathbf{U}_{\widetilde{\mathbf{L}}}^T\mathbf{U}\mathbf{Q}^T\right) = \sum_{i=1}^{k}\lambda_{\widetilde{\mathbf{L}},i}$$

Where we have introduced the unitary diagonalization $\widetilde{\mathbf{L}} = \mathbf{U}_{\widetilde{\mathbf{L}}}\mathbf{S}_{\widetilde{\mathbf{L}}}\mathbf{U}_{\widetilde{\mathbf{L}}}^T$, which can be done since $\widetilde{\mathbf{L}}$ is symmetric too. Moreover, since the ordering of eigenvalues and eigenvectors in a unitary diagonalization is a degree of freedom, we will assume that they are arranged so that the first k correspond to the eigenvectors in the k columns of $\mathbf{U}$. With that arrangement, we can see that $\mathbf{U}^T\mathbf{U}_{\widetilde{\mathbf{L}}} = \left[\mathbf{I}_{k\times k}\ \mathbf{0}_{k\times(N-k)}\right]$. Using that, and the fact that $\mathbf{Q}$ is an orthonormal matrix, we end up with the following fundamental result: the value of the objective function when $\widetilde{\mathbf{H}} = \mathbf{U}\mathbf{Q}^T$ is the sum of the eigenvalues corresponding to the k eigenvectors of $\widetilde{\mathbf{L}}$ in the columns of $\mathbf{U}$.

Point (a) ensures that $\widetilde{\mathbf{H}} = \mathbf{U}\mathbf{Q}^T$ is a valid solution. Finally, point (b) shows that, if we take the k eigenvectors associated to the k smallest eigenvalues of $\widetilde{\mathbf{L}}$, we actually minimize the objective function. Since $\widetilde{\mathbf{H}} = \mathbf{\Pi}^{1/2}\mathbf{H}$, we end up with $\mathbf{H} = \mathbf{\Pi}^{-1/2}\mathbf{U}\mathbf{Q}^T$. □

Note that the transposition of $\mathbf{Q}$ in the expression of $\mathbf{H}$ is merely a notational issue and can be dropped. Moreover, as far as the solution of the *relaxed* problem is concerned, we can completely drop $\mathbf{Q}$ because $\mathbf{Q} = \mathbf{I}$ is a perfectly valid choice. However, as we shall see later, some *rounding* schemes take advantage of the degree of freedom $\mathbf{Q}$ provides to optimize its value, making $\mathbf{H}$ closer to fulfill the non-relaxed constraint $\mathbf{H} = \mathbf{E}\mathbf{\Psi}$. Therefore, we decided to include it in the formulation.

At this point, it is useful to stop for a while and check whether all we have got so far makes sense.

According to the concept of *relaxing* PCut, and using the results of theorem 3, we have that the eigenvectors of the normalized graph Laplacian matrix $\widetilde{\mathbf{L}}$ scaled by $\mathbf{\Pi}^{-1/2}$ should provide very valuable information about the partition which optimizes PCut. As particular cases:

RatioCut: When the target is optimizing RatioCut, $\mathbf{\Pi} = \mathbf{I}$ so that $\widetilde{\mathbf{L}} = \mathbf{L}$. Then, $\mathbf{H}$ consists of the k eigenvectors of the unnormalized graph Laplacian associated to the k smallest eigenvalues. Because of this, we say that RatioCut leads to unnormalized spectral clustering.

NormalizedCut: If we try to optimize NormalizedCut, $\mathbf{\Pi} = \mathbf{D}$ so that $\widetilde{\mathbf{L}} = \mathbf{L}_{sym}$. Besides, (see [2]) if $\mathbf{u}$ is an eigenvector of $\mathbf{L}_{sym}$, $\mathbf{D}^{-1/2}\mathbf{u}$ is both an eigenvector of $\mathbf{L}_{rw}$ and a generalized eigenvector which fulfills $\mathbf{Lu} = \lambda\mathbf{Du}$. Therefore, $\mathbf{H}$ consists of the k eigenvectors of the normalized graph Laplacian $\mathbf{L}_{rw}$ associated to the k smallest eigenvalues.

Does that makes sense? In order to realize that it does, let us consider the simplest scenario we can think of: one in which we have k connected components in the graph. In such a case, we obviously would like that the clustering algorithms identifies each connected component with a cluster. However, thanks to the properties of graph Laplacians, in this scenario 0 is an eigenvalue of $\mathbf{L}$, $\mathbf{L}_{sym}$ and $\mathbf{L}_{rw}$ with multiplicity k, and the corresponding eigenvectors are the indicator vectors of the k connected components (in the case of $\mathbf{L}_{sym}$ the indicator vectors are scaled by $\mathbf{D}^{1/2}$. Therefore, in the ideal case, the solution of the *relaxed* problem $\mathbf{H}$ retains all the information we want.

Now, for more complicated scenarios where the clusters are not so obvious, we can justify that, as long as there are actually several (possible connected) groups of data, the weighted adjacency matrix of the graph will be a slightly perturbed version of the weighted adjacency matrix of the ideal case and, therefore, the eigenvectors of the graph Laplacians will be slightly perturbed versions of the optimal indicator vectors. Indeed, this is one of the multiple interpretations of spectral clustering we will study later.

A very nice example to illustrate due to [2] is as follows. They build a dummy data set consisting of 200 points in $\mathbb{R}$ (one dimensional points) drawn from a mixture of four Gaussians. They construct both a fully-connected graph and a 10-nearest neighbor graph weighting the edges according to a Gaussian similarity function with $\sigma = 1$.

In the upper left corner of figure 4.16 we see the histogram of the data set, which reveals that the data is indeed generated by a mixture of four Gaussian PDFs. The rest of the figure is organized as follows. There are for rows which show a plot of the eigenvalues of a graph Laplacian matrix, and then five plots showing, for each of the first five eigenvectors, the curve x_i vs. $\mathbf{u}_j^i$. The first row considers a 10-nearest neighbor graph with $\mathbf{L}_{rw}$ as graph Laplacian. The second row changes $\mathbf{L}_{rw}$ by $\mathbf{L}$. The third row uses again $\mathbf{L}_{rw}$, but constructed with a fully-connected graph. Finally, the last row uses a fully-connected graph with the unnormalized graph Laplacian $\mathbf{L}$.

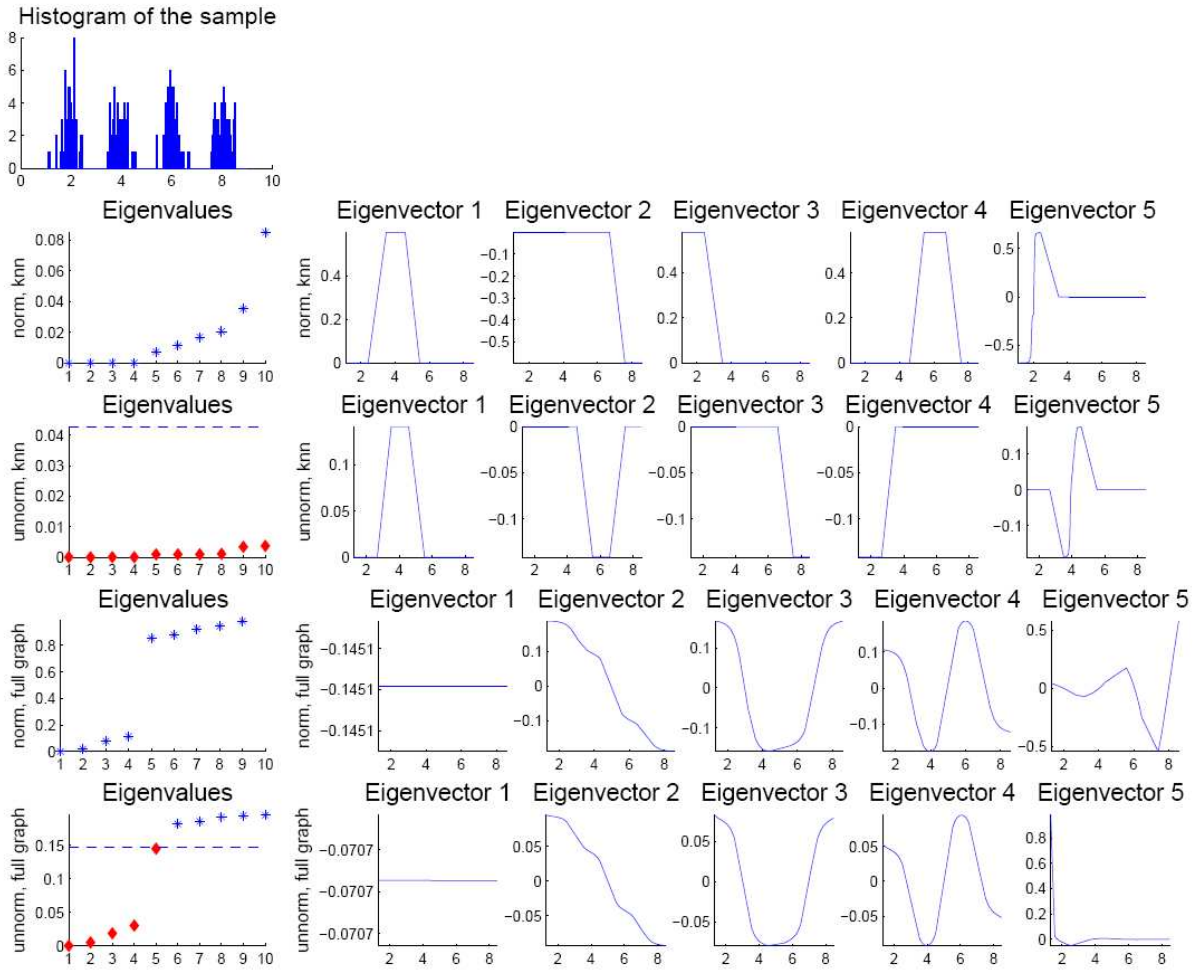


Figure 4.16: Illustration of spectral clustering for a simple data set. See text for details. Image and example taken from [2].

The last five plots in each row are very meaningful because, according to the intuition we just discussed, where the i-th eigenvector with $i = 1, \ldots, k$ is roughly approximation of the indicator vector of the i-th cluster, then the i-th coordinate of the j-th eigenvector shall contain valuable information about whether point $\mathbf{x}_i$ belongs to the j-th cluster or not. Actually, this statement is a great oversimplification of reality. At some point in our way towards achieving the *spectral relaxation*, we introduced $\mathbf{H} = \mathbf{E}\mathbf{\Psi}$. This actually means that, even in the ideal case in which *rounding* is perfect, the columns of $\mathbf{H}$ will not be exactly the indicator vectors but, rather, linear combinations of them. In other words, the eigenvectors of the graph Laplacian are really meaningful, but the information about the indicator vectors in $\mathbf{H}$ is not so overwhelmingly obvious as we would like.

For instance, if we take a look at the first row, we see that the first eigenvector is actually the indicator vector of the second cluster; the third eigenvector, the indicator vector of the first cluster and the forth eigenvector, the indicator vector of the third cluster. However, the second

eigenvector a linear combination of the indicator vectors of clusters 1, 2 and 3. Nevertheless, if we gather the information of the first four eigenvectors, we can easily distinguish the four clusters.

Comparing the nearest neighbor graph with the fully connected graph, we can see that, in this example, using the 10-nearest neighbor graph results in the ideal situation in which we have four connected components (one per cluster). We can see that by realizing that eigenvalue 0 has multiplicity 4, and also from the fact that the eigenvectors are exact linear combinations of the indicator vectors.

However, in the fully-connected graph, we have a certain perturbation. Eigenvalue 0 now has multiplicity 1, something obvious since the graph is fully connected, and the corresponding eigenvector is a constant vector (multiply of $\mathbf{1}$). However, without much effort, we can still spot that the first four eigenvectors allow to differentiate the four clusters quite well, even if now it is not as clear as before.

4.4.4 Rounding schemes

Looking at the results for the fully-connected graph, we have a perfect example of the necessity of *rounding*: the information about the clusters clearly is with the eigenvectors of the graph Laplacians, but we need to further processing of the solution of the *relaxed* problem to obtain the partitions.

There exist many spectral clustering algorithms which mainly vary on three different aspects: the choice of similarity function and the way of constructing the graph, the type of graph Laplacian used and, finally, the *rounding* scheme.

We have already studied some of the main possibilities for the two first degrees of freedom. Now we shall enunciate two of the most typical *rounding* schemes in the literature:

K-means rounding: The most used approach, because of its simplicity and good performance, is to use the K-means clustering algorithm (or any of its variants like K-medians, bisecting K-means or weighted K-means) over the rows of $\mathbf{H}$.

In the ideal case with k connected components, the rows of $\mathbf{H}$ always are one of the k vectors in the canonical basis of $\mathbb{R}^k$, since a point can belong to only cluster in partitional clustering. For all points the j-th cluster, the corresponding row of $\mathbf{H}$ will be the j-th vector of the canonical basis (all its entries 0 except the j-th one). It is obvious that in such an scenario, K-means would converge to the right clusters.

However, for more complicated cases like the ones shown in figure 4.16, the reader can observe that K-means rounding would still converge to the good result.

Non-maximum suppression rounding: An alternative way of rounding the result of the relaxed *problem* is to allocate the i-th data point to the cluster satisfying corresponding to the maximum entry in the i-th row of $\mathbf{H}$, that is, the i-th data point is allocated to the cluster $j = \max_{1 \leq l \leq k} (\mathbf{H})^{i,l}$.

This rounding scheme makes perfect sense too, since by a formal perturbation argument, we expect the entry corresponding to the cluster where the point should belong to be big and the rest small (in the ideal case with k connected components, only the entry corresponding to the connected component to which the point belongs is non-zero).

Procrustean rounding: Procrustean rounding was introduced in [3] as a way of taking advantage of the degree of freedom which exists in the solution of the *relaxed* problem due to the orthonormal matrix $\mathbf{Q}$.

The solution was expressed as $\mathbf{H} = \mathbf{\Pi}^{-1/2}\mathbf{U}\mathbf{Q}^T$. We can try to improve the traditional rounding scheme based on K-means by using an iterative procedure in which we first estimate the value of $\mathbf{E}$ from the current value of $\mathbf{H}$ and, then, we update the value of $\mathbf{Q}$ so that we solve the following Procrustes problem ([29]):

$$\min_{\mathbf{Q}\in\mathbb{R}^{k\times k}} \operatorname{Tr}\left(\left(\mathbf{E}-\mathbf{U}\mathbf{Q}^T\right)\left(\mathbf{E}-\mathbf{U}\mathbf{Q}^T\right)^T\right) = \left\|\mathbf{E}-\mathbf{U}\mathbf{Q}^T\right\|_F^2 \quad s.t. \quad \mathbf{Q}^T\mathbf{Q} = \mathbf{I}_k$$

In other words, we try to force the solution of the *relaxed* problem, $\mathbf{H}$, to be closer to the current cluster-assignation matrix $\mathbf{E}$ (in the sense of minimizing the Frobenius norm) by varying the value of $\mathbf{Q}$. As usual, to solve the constrained optimization problem, we refer to the associated Lagrangian:

$$\mathcal{L}\left(\mathbf{Q},\mathbf{\Phi}\right) = \operatorname{Tr}\left(\left(\mathbf{E}-\mathbf{U}\mathbf{Q}^T\right)\left(\mathbf{E}-\mathbf{U}\mathbf{Q}^T\right)^T\right) + \operatorname{Tr}\left(\mathbf{\Phi}\left(\mathbf{Q}^T\mathbf{Q}-\mathbf{I}_k\right)\right)$$

Where we have introduced a symmetric $k\times k$ matrix of Lagrange multipliers, $\mathbf{\Phi}$. Differentiating with respect to $\mathbf{Q}$ we obtain:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{Q}^T} = 2\mathbf{\Phi}\mathbf{Q}^T - 2\mathbf{U}^T\mathbf{E}$$

Therefore we have that:

$$\mathbf{\Phi}\mathbf{Q}^T = \mathbf{U}^T\mathbf{E} \iff \mathbf{Q}^T = \mathbf{\Phi}^{-1}\mathbf{U}^T\mathbf{E}$$

The constraint $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_k$ is then equivalent to:

$$\mathbf{\Phi}^{-1}\mathbf{U}^T\mathbf{E}\left(\mathbf{U}^T\mathbf{E}\right)^T\mathbf{\Phi}^{-1} = \mathbf{I}_k \iff \mathbf{U}^T\mathbf{E}\left(\mathbf{U}^T\mathbf{E}\right)^T = \mathbf{\Phi}^2$$

We can now introduce the SVD (Singular Value Decomposition) of $\mathbf{U}^T\mathbf{E}$ as $\mathbf{U}^T\mathbf{E} = \mathbf{A}\mathbf{S}\mathbf{B}^T$, with $\mathbf{S}$ being a diagonal matrix with the singular values and $\mathbf{A}$, $\mathbf{B}$ orthonormal matrices whose columns are the left singular vectors and the right singular vectors, respectively. Using that we can rewrite:

$$\mathbf{\Phi}^2 = \mathbf{A}\mathbf{S}^2\mathbf{A}^T$$

From which we can find:

$$\mathbf{\Phi} = \mathbf{ASA}^T \iff \mathbf{\Phi}^{-1} = \mathbf{AS}^{-1}\mathbf{A}^T$$

And, finally, substituting the value of $\mathbf{\Phi}^{-1}$ in $\mathbf{Q}^T = \mathbf{\Phi}^{-1}\mathbf{U}^T\mathbf{E}$ we get:

$$\mathbf{Q}^T = \mathbf{AB}^T$$

Therefore, the final solution is that the value of $\mathbf{Q}^T$ which brings $\mathbf{H}$ closer to the current partition $\mathbf{E}$ can be obtained from the SVD of $\mathbf{U}^T\mathbf{E} = \mathbf{ASB}^T$ as $\mathbf{Q}^T = \mathbf{AB}^T$. An example of an spectral clustering algorithm using the Procrustean rounding approach is in algorithm 4.10.

Since the rounded solution is a heuristically interpreted version of a problem which is a relaxed version of the original objective function, the main disadvantage of spectral clustering is that we lose all theoretical guarantees on the quality of the solution. Indeed, there exist some pathological cases for which it has been shown that spectral clustering converges to a local minimum of PenalizedCut, instead of the global minimum. However, on average, the performance of spectral clustering is excellent and surpasses all of the traditional algorithms. Because of this, finding theoretical results which prove the reasons for such a performance remains a very interesting open problem.

4.4.5 Some spectral clustering algorithms

At this point, we are ready to summarize some of the most well-known variants of spectral clustering. As we said in the previous sections, all of them have basically the structure shown in 4.4.

Particularizing typical choices for the degrees of freedom, we get some of the following variants:

Algorithm 4.5 is probably the most basic form of spectral clustering. It uses the unnormalized graph Laplacian matrix, so that it implements a RatioCut objective function. Also, the rounding scheme is very simple, being based on K-means.

In [4], the authors take advantage of the fact that the generalized eigenvectors obtained through the generalized eigenproblem $\mathbf{Lu}_i = \lambda_i\mathbf{Du}_i$ are identical to those of $\mathbf{L}_{rw}$. Therefore, algorithm 4.6 is actually the same algorithm as 4.5, but implementing NormalizedCut instead of RatioCut.

The main particularity of algorithm 4.7 is the row normalization step. To understand why such a procedure is necessary in this case, we have to refer to the proof of theorem 3. In this algorithm, the authors decided to work with the eigenvectors of $\mathbf{L}_{sym}$. However, according to theorem 3, doing that implies that the eigenvectors we work with are a scaled version of the

Algorithm 4.4: Outline of spectral clustering algorithms

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \mid \mathbf{x}_i \in \mathbb{R}^d$
input : Desired number of clusters k
output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of $\mathbb{X}$

1 Construct a similarity graph as described in section 4.4.2 and compute its weighted adjacency matrix $\mathbf{W}$;
2 Compute one of the possible graph Laplacian matrices $\widetilde{\mathbf{L}}$ depending on the choice of normalization and $\mathbf{W}$;
3 Obtain the k eigenvectors of $\widetilde{\mathbf{L}}$ associated to the k smallest eigenvalues. Store them in a matrix $\mathbf{H}$;
4 Apply a rounding scheme as described in section 4.4.4 to $\mathbf{H}$ to get the cluster assignation matrix $\mathbf{E}$;
5 Translate $\mathbf{E}$ into a partition $\mathbb{S}$ of the input data set.

Algorithm 4.5: Unnormalized spectral clustering ([2])

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \mid \mathbf{x}_i \in \mathbb{R}^d$
input : Desired number of clusters k
output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of $\mathbb{X}$

1 Construct a similarity graph as described in section 4.4.2 and compute its weighted adjacency matrix $\mathbf{W}$;
2 Compute the unnormalized graph Laplacian matrix $\mathbf{L}$ from $\mathbf{W}$;
3 Obtain the k eigenvectors of $\mathbf{L}$ associated to the k smallest eigenvalues, $\mathbf{u}_1, \ldots, \mathbf{u}_k$. Store them by columns in a matrix $\mathbf{H} = [\mathbf{u}_1, \ldots, \mathbf{u}_k] \in \mathbb{R}^{N \times k}$;
4 Cluster the N k-dimensional vectors given by the rows of $\mathbf{H}$, $\{(\mathbf{H})_i\}_{i=1}^N$ into clusters $\mathbb{C}_1, \ldots, \mathbb{C}_k$ using K-means algorithm ;
5 The output partition $\mathbb{S}$ is such that $\mathbb{S}_i = \{\mathbf{x}_j \mid (\mathbf{H})_j \in \mathbb{C}_i\}$;

optimal ones, as $\widetilde{\mathbf{H}} = \mathbf{D}^{1/2}\mathbf{H}$. This is equivalent to saying that each row is scaled by $\sqrt{d_i}$ with respect to the optimal solution of the *relaxed* problem.

There is even a more fundamental need for the row normalization step in 4.7. If we think again about the ideal scenario with k connected components, the eigenvectors of $\mathbf{L}_{sym}$ are not the indicator vectors but rather, scaled versions $\mathbf{D}^{1/2}\mathbf{e}_i$. If a certain point i in the data set has a very low degree, the premultiplication by $\mathbf{D}^{1/2}$, which scales the i-th coordinate of all eigenvectors by d_i, may "mask" the information regarding that point, since the whole row would have values close to zero.

The row normalization step helps overcoming both problems at the same time.

In algorithm 4.8, we show a somewhat simplified version of the algorithm proposed in [6]. The main novelty is regarding the rounding scheme. They use non-maximum suppression rounding

Algorithm 4.6: Normalized spectral clustering ([4])

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \mid \mathbf{x}_i \in \mathbb{R}^d$
input : Desired number of clusters k
output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of $\mathbb{X}$

1 Construct a similarity graph as described in section 4.4.2 and compute its weighted adjacency matrix $\mathbf{W}$;
2 Compute the unnormalized graph Laplacian matrix $\mathbf{L}$ from $\mathbf{W}$;
3 Obtain the k generalized eigenvectors of $\mathbf{L}$ associated to the k smallest generalized eigenvalues $\mathbf{u}_1, \ldots, \mathbf{u}_k$ by solving the generalized eigenproblem $\mathbf{L}\mathbf{u}_i = \lambda_i \mathbf{D}\mathbf{u}_i$. Store them by columns in a matrix $\mathbf{H} = [\mathbf{u}_1, \ldots, \mathbf{u}_k] \in \mathbb{R}^{N \times k}$;
4 Cluster the N k-dimensional vectors given by the rows of $\mathbf{H}$, $\{(\mathbf{H})_i\}_{i=1}^N$ into clusters $\mathbb{C}_1, \ldots, \mathbb{C}_k$ using K-means algorithm ;
5 The output partition $\mathbb{S}$ is such that $\mathbb{S}_i = \{\mathbf{x}_j \mid (\mathbf{H})_j \in \mathbb{C}_i\}$;

Algorithm 4.7: Normalized spectral clustering ([5])

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \mid \mathbf{x}_i \in \mathbb{R}^d$
input : Desired number of clusters k
output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of $\mathbb{X}$

1 Construct a similarity graph as described in section 4.4.2 and compute its weighted adjacency matrix $\mathbf{W}$;
2 Compute the normalized graph Laplacian matrix $\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-/2}$ from $\mathbf{W}$;
3 Obtain the k eigenvectors of $\mathbf{L}_{sym}$ associated to the k smallest eigenvalues, $\mathbf{v}_1, \ldots, \mathbf{v}_k$. Store them by columns in a matrix $\widetilde{\mathbf{H}} = [\mathbf{v}_1, \ldots, \mathbf{v}_k] \in \mathbb{R}^{N \times k}$;
4 Construct a matrix $\mathbf{G} = \in \mathbb{R}^{N \times k}$ by normalizing the rows in $\widetilde{\mathbf{H}}$ to have unit norm, that is, $(\mathbf{G})^{i,j} = (\widetilde{\mathbf{H}})^{i,j} / \left(\sum_{l=1}^k (\widetilde{\mathbf{H}})^{i,l}\right)^{1/2}$;
5 Cluster the N k-dimensional vectors given by the rows of $\mathbf{G}$, $\{(\mathbf{G})_i\}_{i=1}^N$ into clusters $\mathbb{C}_1, \ldots, \mathbb{C}_k$ using K-means algorithm ;
6 The output partition $\mathbb{S}$ is such that $\mathbb{S}_i = \{\mathbf{x}_j \mid (\mathbf{G})_j \in \mathbb{C}_i\}$;

Algorithm 4.8: Simplified version of normalized spectral clustering ([6])

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \mid \mathbf{x}_i \in \mathbb{R}^d$
input : Desired number of clusters k
output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of $\mathbb{X}$

1 Construct a similarity graph as described in section 4.4.2 and compute its weighted adjacency matrix $\mathbf{W}$;
2 Compute the normalized graph Laplacian matrix $\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-/2}$ from $\mathbf{W}$;
3 Obtain the k eigenvectors of $\mathbf{L}_{sym}$ associated to the k smallest eigenvalues, $\mathbf{v}_1, \ldots, \mathbf{v}_k$. Store them by columns in a matrix $\widetilde{\mathbf{H}} = [\mathbf{v}_1, \ldots, \mathbf{v}_k] \in \mathbb{R}^{N \times k}$;
4 Construct a matrix $\mathbf{G} = \in \mathbb{R}^{N \times k}$ by normalizing the rows in $\widetilde{\mathbf{H}}$ to have unit norm, that is, $(\mathbf{G})_{i,j} = (\widetilde{\mathbf{H}})_{i,j} / \left(\sum_{l=1}^{k} (\widetilde{\mathbf{H}})_{i,l}\right)^{1/2}$;
5 Cluster the N k-dimensional vectors given by the rows of $\mathbf{H}$, $\{(\mathbf{H})_i\}_{i=1}^{N}$ into clusters $\mathbb{C}_1, \ldots, \mathbb{C}_k$ using non-maximum suppression rounding ;
6 The output partition $\mathbb{S}$ is such that $\mathbb{S}_i = \{\mathbf{x}_j \mid (\mathbf{H})_j \in \mathbb{C}_i\}$;

as defined in section 4.4.4 over a row-normalized matrix obtained in the same way as in algorithm 4.7.

Algorithm 4.9: PenalizedCut spectral clustering with K-means rounding ([3])

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \mid \mathbf{x}_i \in \mathbb{R}^d$
input : Desired number of clusters k
input : A diagonal matrix of non-negative weights, $\mathbf{\Pi}$
output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of $\mathbb{X}$

1 Construct a similarity graph as described in section 4.4.2 and compute its weighted adjacency matrix $\mathbf{W}$;
2 Compute the normalized graph Laplacian matrix $\widetilde{\mathbf{L}} = \mathbf{\Pi}^{-1/2}\mathbf{L}\mathbf{\Pi}^{-1/2}$ from $\mathbf{W}$;
3 Obtain the k eigenvectors of $\widetilde{\mathbf{L}}$ associated to the k smallest eigenvalues, $\mathbf{u}_1, \ldots, \mathbf{u}_k$. Store them by columns in a matrix $\widetilde{\mathbf{H}} = [\mathbf{u}_2, \ldots, \mathbf{u}_k] \in \mathbb{R}^{N \times (k-1)}$;
4 Compute $\mathbf{H} \in \mathbb{R}^{N \times (k-1)}$ as $\mathbf{H} = \mathbf{\Pi}^{-1/2}\widetilde{\mathbf{H}}$;
5 Cluster the N $(k-1)$-dimensional vectors given by the rows of $\mathbf{H}$, $\{(\mathbf{H})_i\}_{i=1}^{N}$ into clusters $\mathbb{C}_1, \ldots, \mathbb{C}_k$ using weighted K-means algorithm with weigths given by diag $(\mathbf{\Pi})$;
6 The output partition $\mathbb{S}$ is such that $\mathbb{S}_i = \{\mathbf{x}_j \mid (\mathbf{H})_j \in \mathbb{C}_i\}$;

In [3], the authors propose two interesting concepts. On the one hand, they argue that keeping the first eigenvector is useless, since it is a constant vector (or a constant vector scaled by $\mathbf{\Pi}^{1/2}$ and, therefore, lacks discriminative information. This idea allows them to find a very interesting connection between spectral clustering and SVM (Support Vector Machines) in a Reproducing Kernel Hilbert Space (RKHS). They also propose using a weighted K-means algo-

rithm for rounding, based on a result from [30] which shows that a weighted version of K-means arises when rounding is formulated as the difference between projection matrices, which they adapt to their non redundant scheme with $k-1$ dimensional vectors.

Algorithm 4.10: PenalizedCut spectral clustering with Procrustean rounding ([3])

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \mid \mathbf{x}_i \in \mathbb{R}^d$
input : Desired number of clusters k
input : A diagonal matrix of non-negative weights, $\mathbf{\Pi}$
output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \dots, \mathbb{S}_k\}$ of $\mathbb{X}$

Construct a similarity graph as described in section 4.4.2 and compute its weighted adjacency matrix $\mathbf{W}$;
Compute the normalized graph Laplacian matrix $\widetilde{\mathbf{L}} = \mathbf{\Pi}^{-1/2}\mathbf{L}\mathbf{\Pi}^{-1/2}$ from $\mathbf{W}$;
Obtain the k eigenvectors of $\widetilde{\mathbf{L}}$ associated to the k smallest eigenvalues, $\mathbf{u}_1, \dots, \mathbf{u}_k$. Store them by columns in a matrix $\mathbf{U} = [\mathbf{u}_2, \dots, \mathbf{u}_k] \in \mathbb{R}^{N\times(k-1)}$;
Define the auxiliary matrix $\mathbf{G} = \left[\mathbf{I}_{k-1} - \frac{1}{k}\mathbf{1}_{k-1}\mathbf{1}_{k-1}^T, -\frac{1}{k}\mathbf{1}_{k-1}\right]^T$;
Initialize the cluster-assignation matrix $\mathbf{E}$;
while *convergence criterion not satisfied* **do**
    Obtain $\mathbf{U}^T\mathbf{E}\mathbf{G} = \mathbf{A}\mathbf{S}\mathbf{B}^T$ through the SVD (Singular Value Decomposition) of the $(k-1)\times(k-1)$ matrix $\mathbf{U}^T\mathbf{E}\mathbf{G}$;
    $\mathbf{Q} \leftarrow \mathbf{A}\mathbf{B}^T$;
    $\mathbf{H} \leftarrow \mathbf{\Pi}^{-1/2}\mathbf{U}\mathbf{Q}$;
    for $i \leftarrow 1$ **to** N **do**
        Assign $\mathbf{x}_i$ to the cluster $j = \begin{cases} k & \text{if } (\mathbf{H})^{i,l} \leq 0 \; \forall\, l = 1, \dots, k-1 \\ \max\limits_{1\leq l\leq k} (\mathbf{H})^{i,l} & \text{otherwise} \end{cases}$;
        $(\mathbf{E})^{i,k} \leftarrow \begin{cases} 1 & \text{if } k = j \\ 0 & \text{otherwise} \end{cases}$;
    end
    Translate $\mathbf{E}$ into a partition $\mathbb{S}$ of the input data set ;
end

Algorithm 4.10 will be the last general purpose spectral clustering algorithm we show in this project. The idea is very similar to that of algorithm 4.9, but the rounding scheme has been changed from weighted K-means to Procrustean rounding. We can see that the Procrustean rounding is implemented by first initialize somehow (randomly, using K-means rounding, maximum element rounding or however we like) the partition matrix $\mathbf{E}$ and, after that, we enter an iterative procedure in which we compute $\mathbf{Q}$ solving the Procrustean problem so that the *relaxed* solution $\mathbf{H}$ is closer to the current $\mathbf{E}$. After recomputing $\mathbf{H}$, they update $\mathbf{E}$ by using maximum element rounding and the procedure is repeated until convergence.

The introduction of matrix $\mathbf{G}$ may seem a bit obscure at first sight. Its inclusion is due to the non-redundant formulation of spectral clustering used by [3]. In fact, we can see that postmultiplying $\mathbf{E}$ by $\mathbf{G}$ acts by "erasing" the information in the last column of $\mathbf{E}$, reducing the size of the new cluster assignation matrix to $N \times (k-1)$, and adding a negative offset of value $-1/k$ to the whole resulting matrix. Note that, actually, we can still recover the information about the k-th cluster since any point which does not belong to any cluster from 1 to $k-1$ must necessarily belong to the k-th cluster. With the offset, in the ideal situation when the columns of $\mathbf{H}$ are indicator vectors of connected components, we should have that the j-th entry in the i-th row of $\mathbf{H}$ has value $\frac{k-1}{k}$ if the i-th data point belongs to cluster j, and value $-\frac{1}{k}$ otherwise. If all the entries of a row are non-positive, then we can argue that the data point associated to that row belongs to the k-th cluster, whose indicator vector was erased. This idea is the one which justifies the Procrustean rounding approach with $\mathbf{G}$ premultiplying $\mathbf{E}$, followed by non-maximum suppression rounding in the rows of the updated $\mathbf{H}$. In other words, it is actually true that the $N \times k$ cluster assignation matrix that we have been using in this section contains redundant information.

Choosing the number of clusters

One of the few disadvantages of spectral clustering is that, just as for K-means, the number of clusters needs to be known a priori.

However, from the discussion in section 4.4.3, we know that the eigenvalues of the chosen graph Laplacian matrix contain useful information regarding the optimal number of clusters. Indeed, in the ideal case in which we have k connected components, the first k eigenvalues are 0. Using the formal perturbation argument, we can expect that more realistic cases have all but one non-zero eigenvalue. Nonetheless, if the clusters are more of less well differentiated as measured by the graph structure and similarity function chosen, then we can expect the first k eigenvalues to be small and to observe a gap between the k-th eigenvalue and the $(k+1)$-th eigenvalue. This idea is usually called "the eigengap heuristic". However, if the graph is very tangled, with lots of connections between clusters with high weights, as it happens when the existence of groups is very ambiguous, this heuristic does not provide clear results and we must fall back to making parametric sweeps.

Other formulations of spectral clustering

One of the fascinating aspects of spectral clustering is the surprising fact that it arises as the solution of a wild variety of clustering criteria which, a priori, have absolutely nothing in common. Readers with further interest in that topic can find a discussion on some of those alternative formulations of spectral clustering in appendix A.

4.5 Mean shift clustering

In 1975, Keinosuke Fukunaga and Larry Hostetler proposed the mean shift algorithm [31], an innovative non-parametric mode seeking procedure. It is a very simple yet powerful algorithm. However, it has received few attention in the specialized literature in statistics or machine learning. Indeed, after the original paper, mean shift was basically forgotten until it was recovered by Cheng in 1995 [32] and popularized by [33] in 2002, after an article where they show the excellent qualities of the algorithm by applying it to computer vision.

Mean shift defines an iterative gradient ascent procedure in order to find the modes (maxima) of the probability density function (PDF) underlying the data set. However, since we do not have access to this probability density, this problem appears to be unsolvable. However, the mean-shift algorithm combines non-parametric PDF estimator theory with the gradient ascent algorithm in order to build a fast, robust non-parametric mode seeking algorithm.

4.5.1 Kernel density estimation: the Parzen window technique

The kernel density estimator, also known as Parzen windows density estimator due to the original author who proposed the method [34], is one of the most popular PDF estimators nowadays. The method follows a non-parametric approach, that is, no assumptions are made on the underlying distribution. Consider a data set with N observations in $\mathbb{R}^d$, $\{\mathbf{x}_i\}_{i=1}^N$, generated by an unknown distribution with density $f(\mathbf{x})$, then the multivariate Parzen window estimator with kernel $K(\mathbf{x})$ is defined as:

$$\hat{f}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} K_{\mathbf{H}_i}(\mathbf{x} - \mathbf{x}_i) \tag{4.44}$$

Where $K_{\mathbf{H}_i}(\mathbf{x})$ is constructed from a d-variate kernel $K(\mathbf{x})$ as:

$$K_{\mathbf{H}_i}(\mathbf{x} - \mathbf{x}_i) = |\mathbf{H}_i|^{-1/2} K\left(\mathbf{H}_i^{-1/2}\mathbf{x}\right) \tag{4.45}$$

With $\mathbf{H}_i$ a symmetric positive-definite $d \times d$ matrix, referred to as the *bandwidth* matrix.

Equation (4.44) can be interpreted as a mixture where each observation corresponds to a different component with associated weight equal to the probability of drawing that sample, estimated as its number of occurrences over the total number of samples, whereas the kernel $K_{\mathbf{H}_i}(\mathbf{x})$ determines the shape of each component. The concept becomes clearer in figure 4.17.

We may be tempted to think that kernel density estimation is a semi-parametric method, where the kernel function $K_{\mathbf{H}_i}(\mathbf{x} - \mathbf{x}_i)$ corresponds to the assumed shape of the underlying PDF. However, there is a fundamental difference: in semi-parametric estimation, we assume the shape of the density and use the data to estimate the parameters of the distribution whereas in kernel density estimation, the summation process guarantees that, as the number of samples in the data set, N, increases, the density estimate $\hat{f}(\mathbf{x})$ will converge to a meaningful function of

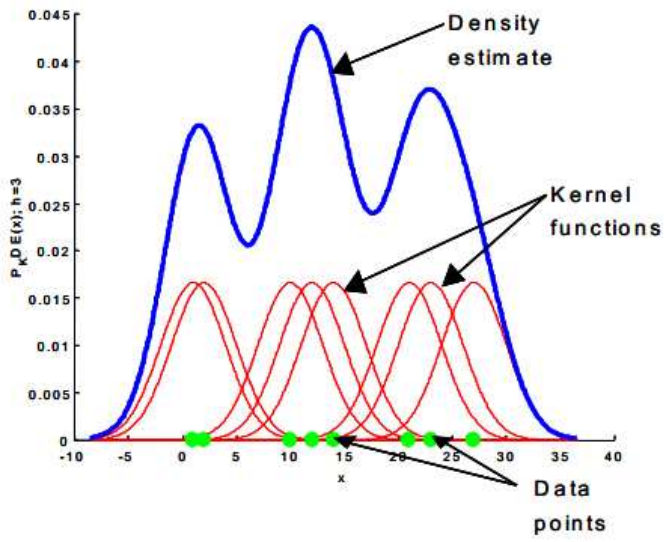


Figure 4.17: Kernel density estimation as a mixture with one component per sample.

the real density $f(\mathbf{x})$ no matter what the choice of $K(\mathbf{x})$ is, as we will discuss later. Therefore, kernel density estimation is a non-parametric method.

Nevertheless the kernel function has a strong effect in the performance of the algorithm. Depending on the particular situation, some kernels are more appropriate than others or, simply, we have to deal with some trade-off between the advantages offered by each type of kernel. Later we will discuss more about different typical choices for kernels and their relation with the performance of the density estimate.

Another fundamental parameter in kernel density estimation is the set of matrices $\mathbf{H}_i \in \mathbb{R}^{d\times d}\ i = 1, \ldots, N$. In order to explain the purpose of $\mathbf{H}_i$ we will begin by considering the most widely used approach in which we set $\mathbf{H}_i = \mathbf{H} = h^2\mathbf{I}_d\ \forall\, i = 1, \ldots, N$. The main reason for this simplification is the great reduction in complexity since we have to tune only one parameter instead of Nd^2. This will also reduce the risk of overfitting, at the price of less power of expression. For problems where an Euclidean-metric performs poorly, we may need to resort to the estimation of the whole matrix.

If we introduce the previous simplification into equation (4.45) we obtain:

$$K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_i) = \frac{1}{h^d} K\left(\frac{\mathbf{x}}{h}\right) \tag{4.46}$$

In this case, we clearly see that the bandwidth parameter acts as a scaling parameter which controls the width of the kernel function. The easiest way to see this is by considering the commonly used uniform kernel:

$$K_U(\mathbf{x}) = \begin{cases} c_d^{-1} & \text{if } \|\mathbf{x}\| \leq 1 \\ 0 & \text{if } \|\mathbf{x}\| > 1 \end{cases} \tag{4.47}$$

Where c_d denotes the volume of the d-dimensional hypersphere of unit radius, $c_d = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}$.

Substituting (4.47) in (4.44) we see that the estimator is built as a superposition of spherical windows $K_U(\mathbf{x})$ of radius h centered at each observation. The basic idea of kernel density estimation is that regions of the sample space where there is a high density of probability will breed many observations in the data set, so that lots of terms in the summation will add a kernel window centered somewhere in the region, whereas regions with a low probability density will provide few samples in the data set and, hence, less summation terms will add up in that area leading to a low value of the estimated density.

However, if we increase h too much, the kernel window becomes too wide and it no longer represents the local information about the density of the observations, since windows centered in high density region would leak onto lower density regions, and the estimation becomes too smooth. On the other hand, by setting h too small, we get an estimate which fails to "fill the gaps" between samples so that it underestimates the support of the probability density function by assuming that the probability density is zero in a lot of places between the observations in the data set. We show a graphical example in figure 4.18.

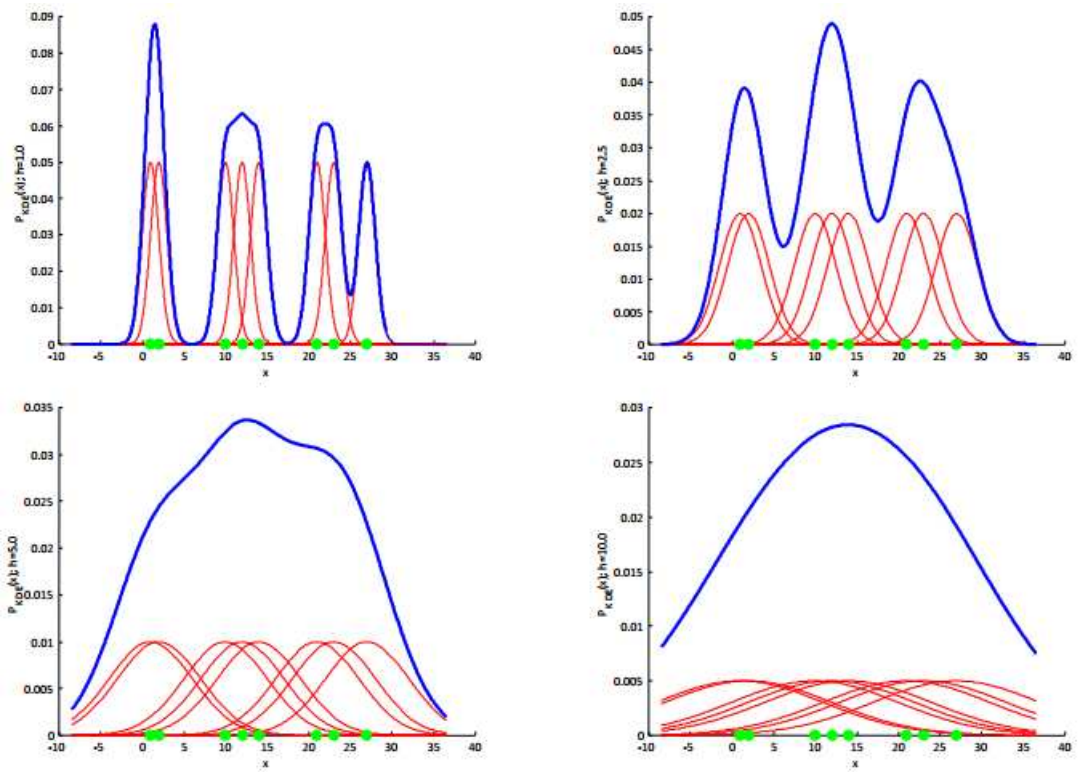


Figure 4.18: Illustration of the effect of different choices for the bandwidth parameter h in a one-dimensional scenario.

To make it even more clear, we can think of the two extreme cases. When $h \to \infty$, the kernel window becomes a constant function and so does the kernel density estimate: it is a case of extreme probability density leakage in the estimation. Alternatively, when $h \to 0$, the

kernel window approximates a Dirac's delta, and the estimator would tell us that the probability density at any point which is not an observation in the data set is 0.

In order to increase the expressive power of the algorithm, at the price of increase complexity, we can act in two directions.

On the one hand, we can allow a more complex representation of the data by using a complete bandwidth matrix $\mathbf{H}$. This is very useful for cases where the Euclidean-metric is not appropriate, like for elliptical clouds with high eccentricity. Indeed, as we will discuss a few paragraphs later, we usually work with kernels $K(\mathbf{x})$ which are actually a function of the $\|x\|^2$. Then using a complete matrix is equivalent to changing from Euclidean-distance to Mahalanobis-distance with covariance matrix $\mathbf{H}$.

On the other hand, it is possible to allow a different bandwidth parameter h_i per sample, or even a different bandwidth matrix $\mathbf{H}_i$. This allows to represent faithfully data composed of mixtures with very different shapes at the price of a very high complexity, with N parameters to be tuned if we use a per-sample scalar bandwidth of as much as Nd^2 parameters where we use a whole per-sample bandwidth matrix.

Apart from manual tuning of the bandwidth parameters by the use of cross-validation, it is possible to use the following heuristic. If radially symmetric kernels are desired, the average distance under some metric (city block, euclidean, supremum, etc) to the K-nearest neighbors gives an estimate of h_i, [35]. On the other hand, if a complete per-sample bandwidth matrix $\mathbf{H}_i$ is to be used, we can estimate the local covariance matrix around the i-th point, $\mathbf{\Sigma}_i$, using the K-nearest neighbors of that point and set $\mathbf{H}_i = \mathbf{\Sigma}_i^{-1}$. More complicated alternatives, like the usage of genetic algorithms for tuning the bandwidth parameters, will be explored in the next chapter for our particular application.

Kernel functions and kernel profiles

The multivariate kernel $K(\mathbf{x})$ can be any bounded function with compact support satisfying the following properties:

Non-negativity: $K(\mathbf{x}) \geq 0$

Normalized: $\int_{\mathbb{R}^d} K(\mathbf{x}) d\mathbf{x} = 1$

Symmetric (even function): $\int_{\mathbb{R}^d} \mathbf{x} K(\mathbf{x}) d\mathbf{x} = 0$

Exponential decay: $\lim_{\|\mathbf{x}\| \to \infty} \|\mathbf{x}\|^d K(\mathbf{x}) = 0$

Uncorrelated: $\int_{\mathbb{R}^d} \mathbf{x}\mathbf{x}^T K(\mathbf{x}) d\mathbf{x} = c_K \mathbf{I}$

With c_K a kernel-dependent scalar.

There are two main ways of generating a multivariate kernel $K(\mathbf{x})$ from a univariate kernel $K_1(x)$.

The first one is motivated by the well-known property of probability density functions which establishes that a multivariate density can be written as the product of the univariate densities in each dimension provided that the distinct features of the random vector are statistically independent. Note that defining a multivariate kernel as the product of univariate kernels does not imply any assumption on the independence of the features. It is just a "motivation" but the properties of the estimator won't depend on that, as long as the resulting multivariate kernel $K(\mathbf{x})$ fulfills all the properties previously listed. Mathematically this is expressed as:

$$K(\mathbf{x}) = \prod_{i=1}^{d} K_1(x_i) \tag{4.48}$$

The other common choice is to generate the multivariate kernel by rotating $K_1(x)$ in $\mathbb{R}^d$, giving raise to a radially symmetric d-variate kernel. Mathematically we can write that idea as:

$$K(\mathbf{x}) = c_{k,d} K_1(\|x\|) \tag{4.49}$$

The introduction of the normalization constant $c_{k,d}$ is due to the fact that:

$$\int_{\mathbb{R}} K_1(x)dx = 1 \implies \int_{\mathbb{R}^d} K_1(\|x\|)d\mathbf{x} = 1 \tag{4.50}$$

Therefore, we must set $c_{k,d}$ in order to ensure that the resulting multivariate kernel actually fulfills the properties of a kernel function. As a technicality, we must point out the normalization requirement for kernels is not really needed in the context of mean-shift. Nevertheless, we will retain it for consistency with the specialized literature.

The reader can check that both types of multivariate kernels will satisfy the five properties of kernels listed before as long as the generating univariate kernel $K_1(x)$ in $\mathbb{R}^d$ fulfills them too. Hence, we can use any of the standard univariate kernels to obtain a d-variate kernel.

Even though both types of kernels are valid, as we previously anticipated, along this project we will focus on radially symmetric kernels which can be written as:

$$K(\mathbf{x}) = c_{k,d} k(\|\mathbf{x}\|^2) \tag{4.51}$$

This is because, in our particular application, we will mainly work with isotropic signal propagation, which is radially symmetric. Moreover, if we needed to add more expressive power to our kernel, it is perfectly possible to add an arbitrary covariance matrix to the previous formulation, allowing us to model also directive antennas. We will discuss that in more detail during the next chapter.

In other words, the kernel can be written as a scalar function of the norm squared of $\mathbf{x}$. If the Euclidean metric induced by this representation was not a good choice for a certain context, the usage of a complete bandwidth matrix $\mathbf{H}$ can help fix the problem, as we previously discussed.

The mapping $k : \mathbb{R}^+ \mapsto \mathbb{R}^+$, denoted the *profile* of the kernel, is the key which controls the characteristics of the resulting multivariate radially symmetric kernel $K(\mathbf{x})$. Apart from being

non-negative, the kernel profile function must be non-increasing, $a < b \implies k(a) \geq k(b)$, and continuous except on possibly a finite set of points. As before, the constant $c_{k,d}$ is a positive scalar calculated to ensure that $K(\mathbf{x})$ has unit energy. The subindex notation refers to the fact that $c_{k,d}$ is the normalization constant for the d-variate kernel induced by the kernel profile $k(x)$.

As an example, the uniform kernel, introduced in equation (4.47), is generated by the so called uniform kernel profile:

$$k_U(\mathbf{x}) = \begin{cases} 1 & \text{if } x \leq 1 \\ 0 & \text{if } x > 1 \end{cases} \tag{4.52}$$

Besides the uniform kernel which was introduced before, another two widely used kernel profiles are, on the one hand, the exponential profile:

$$k_N(x) = e^{-\frac{1}{2}x} \tag{4.53}$$

Which generates the most-well multivariate kernel, the Gaussian or Normal kernel:

$$K_N(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d}} e^{-\frac{1}{2}\|x\|^2} \tag{4.54}$$

Mainly because of implementation issues, it is frequent to define a truncated Normal kernel:

$$K_{N,T}(\mathbf{x}) = \begin{cases} c_{k,d} e^{-\frac{1}{2}\|x\|^2} & \text{if } \|x\| \leq T \\ 0 & \text{if } \|x\| > T \end{cases} \tag{4.55}$$

Where T is an arbitrary threshold which choses from where on we truncate the kernel. Note that, being rigorous, the truncation will force us to estimate the normalization constant $c_{k,d}$ using numerical methods.

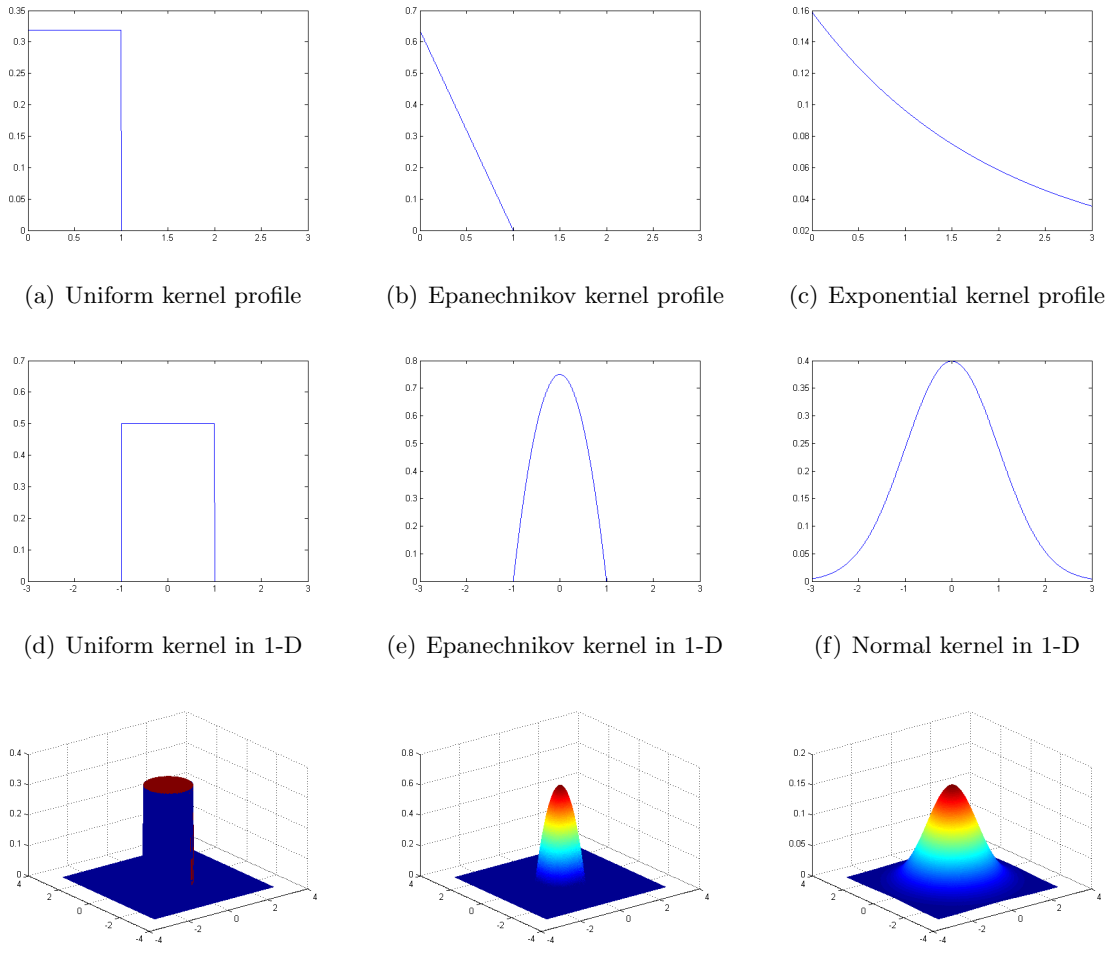
And, in the other hand, the Epanechnikov profile:

$$k_E(x) = \begin{cases} 1 - x & \text{if } 0 \leq x \leq 1 \\ 0 & \text{if } x > 1 \end{cases} \tag{4.56}$$

Which gives raise to the Epanechnikov kernel:

$$K_E(\mathbf{x}) = \begin{cases} \frac{1}{2} c_d^{-1} (d+2)(1 - \|x\|^2) & \text{if } \|x\| \leq 1 \\ 0 & \text{if } \|x\| > 1 \end{cases} \tag{4.57}$$

For both cases, the constants shown in the expressions of the multivariate kernels are to ensure they integrate to unity, where c_d was previously defined as the volume of the unit-hypersphere in $\mathbb{R}^d$.



(a) Uniform kernel profile (b) Epanechnikov kernel profile (c) Exponential kernel profile

(d) Uniform kernel in 1-D (e) Epanechnikov kernel in 1-D (f) Normal kernel in 1-D

(g) Uniform kernel in 2-D (h) Epanechnikov kernel in 2-D (i) Normal kernel in 2-D

Figure 4.19: Graphical illustration of kernel profiles and their corresponding univariate and bivariate kernels with $\mathbf{H} = \mathbf{I}_d$.

Principle behind Parzen window density estimation

A very nice intuition on how kernel density estimation works can be obtained from a non-rigorous study of its asymptotic behavior. Let us assume that the number of samples in the data set is very big, so that we may approximate $N \to \infty$. If that's the case, we can assume that we will have samples in the data set all over the support of the probability density function generating the data. Moreover, we know that, following a frequentist approach, we will have approximately $Nf(\mathbf{x}_i)d\mathbf{x}_i$ samples in a neighborhood with differential volume $d\mathbf{x}_i$ around point $\mathbf{x}_i$. Following this observation, we see that equation (4.44) becomes:

$$\lim_{N\to\infty} \hat{f}(\mathbf{x}) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_i) = \int_{\mathbb{R}^d} K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_i) f(\mathbf{x}_i) d\mathbf{x}_i \tag{4.58}$$

To avoid mathematicians' terror, we must insist in the fact that this a very non-rigorous analysis. Convergence of the previous limit has to be understood in the sense of convergence of distributions (generalized functions). Indeed, we did not even prove at any point that the limit exists, and it is not obvious at all that it does. Nevertheless, a rigorous analysis is completely outside of the scope of this project. Here, we are content with providing a simple intuitive analysis.

Equation (4.58) is simply saying that a kernel density estimator converges asymptotically to the convolution between the kernel $K_{\mathbf{H}}(\mathbf{x})$ and the real probability density function $f(\mathbf{x})$. This is a really nice result, especially for people familiar with the topic of signal processing, since it can rephrased as stating that the kernel density estimator converges asymptotically to the the real probability density function filtered by the multivariate kernel $K_{\mathbf{H}}(\mathbf{x})$.

As an example, this problem is fully equivalent to the estimation of the spectrum of a signal from a finite set of samples. Depending on the weighting function applied to the samples, the so called *windows*, we get a different estimation in which the real spectrum of the signal gets filtered by a filter whose impulse response equals the spectrum of the window. Different choices for window function such as rectangular windows, triangular windows, Hamming, Hanning or Cosine allow the engineer to find a trade-off between resolution and spectral leakage due to the side lobes.

Here, we have exactly the same problem. Depending on the choice of the kernel, we will get some distortion in the estimate, From a statistical point of view, this implies that the estimator we have derived is biased. For instance, if the kernel function is wide, two modes which are very close can appear to be a single mode according to the estimator. This is fully equivalent to the resolution limit which appears when looking for harmonics of a signal from some discrete-time samples stored in memory. Similarly, a mode with a very high probability density can mask other nearby modes with a lower density if the kernel function has significant side lobes, which is equivalent to the problematic of spectral leakage for spectral analysis of signals. We can represent that in block diagram form as in figure 4.20. A graphical example is provided in figure 4.21, where we can see that two modes have been wrongly identified as one due to an unfortunate choice of the kernel parameter h.

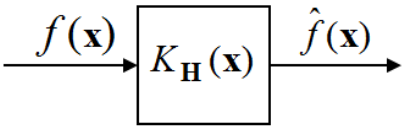
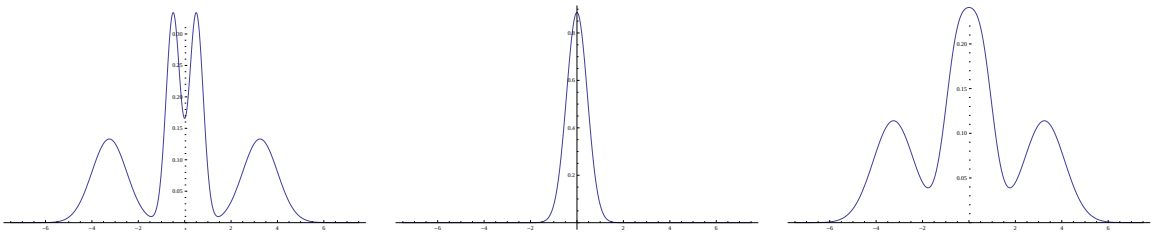


Figure 4.20: Block diagram representation of asymptotic behavior of kernel density estimators.

In particular, for the estimator to be asymptotically unbiased, we require that the bandwidth matrix $\mathbf{H}$ is chosen in such a way that:

$$\lim_{N\to\infty} K_{\mathbf{H}}(\mathbf{x}) = \delta(\mathbf{x}) \tag{4.59}$$



(a) Original PDF with four modes (b) Normal univariate kernel with $h = 0.45$ (c) Asymptotic kernel density estimate

Figure 4.21: The kernel density estimator converges asymptotically to the underlying probability density function of the data set filtered by the kernel function.

Where $\delta(\mathbf{x})$ denotes the Dirac delta function. Again, the previous statement is actually "playing with fire" and it must be handled with extreme caution. However, for a non-rigorous analysis it is enough.

It can be proven (see [31]) that the condition to obtain asymptotic unbiasedness is to choose the bandwidth of the kernel h such that $\lim_{N\to\infty} h(N) = 0$. Moreover, mean square consistency is obtained if $\lim_{N\to\infty} Nh^d(N) = \infty$ and uniform consistency (in probability) if $\lim_{N\to\infty} Nh^{2d}(N) = \infty$. In that work, the author employed the common simplification $\mathbf{H}_i = h^2\mathbf{I}_d \; \forall \; i = 1, \ldots, N$ as we previously discussed.

In the previous analysis, we assumed that all samples use the same bandwidth matrix, $\mathbf{H}_i = \mathbf{H} \; \forall \; i = 1, \ldots, N$. That can be generalized for an arbitrary set $\{\mathbf{H}_i\}_{i=1}^N$. Since we are assuming $N \to \infty$, we need to define a continuous function $\mathbf{H}_i = \mathbf{H}(\mathbf{x})$ so that the convolution becomes lightly more complex:

$$\lim_{N\to\infty} \hat{f}(\mathbf{x}) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} K_{\mathbf{H}_i}(\mathbf{x} - \mathbf{x}_i) = \int_{\mathbb{R}^d} K_{\mathbf{H}(\mathbf{x}_i)}(\mathbf{x} - \mathbf{x}_i) f(\mathbf{x}_i) d\mathbf{x}_i = \\ = \int_{\mathbb{R}^d} K(\mathbf{x} - \mathbf{x}_i, \mathbf{H}(\mathbf{x}_i)) f(\mathbf{x}_i) d\mathbf{x}_i \tag{4.60}$$

This is equivalent to a time-varying convolution, similar to the one which arises in the study of behavior of communication channels with a time-varying impulse response. Here, the "impulse response" is "time-dependent" since for each value of "time" $\mathbf{x}_i$ the bandwidth changes. Similar analogies can be found for convolutions with a spatial meaning or in more complex scenarios.

Kernel density estimates are compared in terms of the mean squared error between the real probability density function and the estimate integrated over the support of the density function (MISE). Since evaluating that criterion is not always feasible analytically, it is frequent that an asymptotic approximation (AMISE) is used instead. It is possible to show, [36], that the Epanechnikov profile minimizes both MISE and AMISE when generating the multivariate kernels as in (4.48) or in (4.49).

The kernels we have shown are by far the most widely used. More importantly, they are the

ones which are interesting for this project, especially the Gaussian kernel. However, the reader which has a further interest on the topic of probability density function estimation can find a wild variety of kernels in the literature, each tailored to particular applications.

4.5.2 PDF gradient estimation

Once we know how to estimate the density function of the data set, we will focus our attention in how to estimate the modes of the distribution. In a first thought, since the modes are defined as the set of maxima of the probability density function, we know they must be located within the set of singular points of the probability density function, that is, those points which satisfy that $\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$. While it is true that we do not know the original underlying distribution of the data, $f(\mathbf{x})$, it is equally undeniable that we did not spend a lot of effort in developing good estimates $\hat{f}(x)$ for nothing. Indeed, we could try to locate the set maxima of the probability density estimator and use them as an estimator for the location of the modes... And that is exactly what we are going to do.

Indeed, the mean shift procedure is nothing more, but nothing less, that a very simple yet extremely elegant way of locating the singular points of the density estimate without actually computing such estimate.

Let us begin from the original expression of the kernel density estimator as in (4.44):

$$\hat{f}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} K_{\mathbf{H}_i}(\mathbf{x} - \mathbf{x}_i) \tag{4.61}$$

Substituting $K_{\mathbf{H}_i}(\mathbf{x}-\mathbf{x}_i) = |\mathbf{H}_i|^{-1/2} K\left(\mathbf{H}_i^{-1/2}\mathbf{x}\right)$ and focusing on kernels of the form $K(\mathbf{x}) = c_{k,d}k(\|\mathbf{x}\|^2)$ we can express the kernel density estimator as:

$$\hat{f}(\mathbf{x}) = \frac{c_{k,d}}{N} \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} k\left(\|\mathbf{x} - \mathbf{x}_i\|^2_{\mathbf{H}_i}\right) \tag{4.62}$$

Where $\|\mathbf{x}\|_{\mathbf{H}_i} = \sqrt{\mathbf{x}^T \mathbf{H}_i^{-1} \mathbf{x}}$ denotes the Mahalanobis norm with covariance matrix $\mathbf{H}_i^{-1}$.

Now, we are going to try to find an estimator of the gradient of the original probability density function of the data, $f(\mathbf{x})$. As we discussed before, the most obvious choice is to build such an estimator as the gradient of the PDF estimate:

$$\hat{\nabla}_{\mathbf{x}} f(\mathbf{x}) = \nabla_{\mathbf{x}} \hat{f}(\mathbf{x}) = \nabla_{\mathbf{x}} \left(\frac{c_{k,d}}{N} \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} k\left(\|\mathbf{x} - \mathbf{x}_i\|^2_{\mathbf{H}_i}\right) \right) \tag{4.63}$$

We can compute the expression in (4.63) exploiting the linearity of the kernel density estimate and applying the chain rule with the quadratic form $\|\mathbf{x} - \mathbf{x}_i\|^2_{\mathbf{H}_i}$, whose gradient is given by $\nabla_{\mathbf{x}} \|\mathbf{x} - \mathbf{x}_i\|^2_{\mathbf{H}_i} = 2\mathbf{H}_i^{-1}(\mathbf{x} - \mathbf{x}_i)$. Then we have that:

$$\hat{\nabla}_{\mathbf{x}} f(\mathbf{x}) = \frac{2c_{k,d}}{N} \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\|\mathbf{x} - \mathbf{x}_i\|^2_{\mathbf{H}_i}\right) \mathbf{H}_i^{-1}(\mathbf{x}_i - \mathbf{x}) \tag{4.64}$$

Where we have defined the function:

$$g(x) = -\frac{dk(x)}{dx} \tag{4.65}$$

Assuming that the kernel profile $k(x)$ is differentiable in $\mathbb{R}^+$ except on possibly a finite set of points. It is very interesting to note that since $k(x)$ is, by definition, a non-increasing continuous function, $g(x)$ actually defines a new kernel profile, which can be used to induce a new multivariate kernel:

$$G(\mathbf{x}) = c_{g,d} g(\|\mathbf{x}\|^2) \tag{4.66}$$

We say that kernel $K(\mathbf{x})$ is the *shadow* of kernel $G(\mathbf{x})$. The reader can check that a set of interesting dualities between the kernels defined before arise. Indeed, the Epanechnikov kernel is the shadow of the uniform kernel and Gaussian functions still exhibit their typical "endogamous" properties, so that the Normal kernel is its own shadow.

4.5.3 The mean-shift algorithm

Equation (4.64) is the basis of mean shift algorithm. We remind the reader that our current objective is no other that finding singular points of $f(\mathbf{x})$ by estimating its gradient $\nabla_x f(\mathbf{x})$ as the gradient of its estimator, $\nabla_x \hat{f}(\mathbf{x})$.

Then, under this approach, we try to find points $\mathbf{x}^*$ satisfying $\nabla_x \hat{f}(\mathbf{x}^*) = 0$. Sadly, solving for such a $\mathbf{x}^*$ is an unsurmountable task except for a few simple kernel profiles. Instead, the clever trick firstly proposed by [31], the mean shift algorithm, is based in the observation that for points $\mathbf{x}^{(t)}$ sufficiently close to a singular point $\mathbf{x}^*$ of $f(\mathbf{x})$, defining the following iteration:

$$\mathbf{x}^{(t+1)} \leftarrow \left(\sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\| \mathbf{x}^{(t)} - \mathbf{x}_i \right\|_{\mathbf{H}_i}^2 \right) \mathbf{H}_i^{-1} \right)^{-1} \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\| \mathbf{x}^{(t)} - \mathbf{x}_i \right\|_{\mathbf{H}_i}^2 \right) \mathbf{H}_i^{-1} \mathbf{x}_i \tag{4.67}$$

Will define a convergent, non-decreasing sequence whose limit can be then used as an estimate for one of the modes of the distribution. Later we will show a proof of that statement.

The most common form used in the literature ([31], [32] or [33] for instance) is to work only with the simplest case, that is, $\mathbf{H} = h^2 \mathbf{I}_d$. It can be readily checked that, under that assumption, equation (4.67) becomes much simpler:

$$\mathbf{x}^{(t+1)} \leftarrow \frac{\sum_{i=1}^{N} \mathbf{x}_i g\left(\left\| \mathbf{x}^{(t)} - \mathbf{x}_i \right\|_{\mathbf{H}_i}^2 \right)}{\sum_{i=1}^{N} g\left(\left\| \mathbf{x}^{(t)} - \mathbf{x}_i \right\|_{\mathbf{H}_i}^2 \right)} \tag{4.68}$$

However, in this project, we stick to the most general case and obtain the well-known results by particularizing $\mathbf{H}_i$.

Nonetheless, it is interesting to note that, if we use the same bandwidth matrix for all samples, that is, $\mathbf{H}_i = \mathbf{H}\ \forall\, i = 1, \ldots, N$, the expression in (4.67) also simplifies into (4.68) even if $\mathbf{H}$ is not of the form $\mathbf{H} = h^2\mathbf{I}_d$.

We can also get a slightly simpler form of (4.67) if we require radially symmetric kernels (a scalar bandwidth) but still allow using different bandwidths per sample, that is, $\mathbf{H} = h_i^2\mathbf{I}_d$. Then, we obtain:

$$\mathbf{x}^{(t+1)} \leftarrow \frac{\sum_{i=1}^{N} \frac{1}{h_i^{d+2}}\mathbf{x}_i g(\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right)}{\sum_{i=1}^{N} \frac{1}{h_i^{d+2}} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right)} \tag{4.69}$$

To get a deeper insight on how mean shift operates, it is very interesting to consider what happens in equation (4.64) when $\mathbf{H} = h^2\mathbf{I}_d$. Indeed, we see that the expression of the gradient estimate simplifies to:

$$\hat{\nabla}_{\mathbf{x}} f(\mathbf{x}) = \frac{2c_{k,d}}{Nh^{d+2}} \sum_{i=1}^{N} (\mathbf{x}_i - \mathbf{x}) g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right) \tag{4.70}$$

Which can be rewritten as:

$$\hat{\nabla}_{\mathbf{x}} f_{K,h}(\mathbf{x}) = \frac{2c_{k,d}}{h^2 c_{g,d}} \hat{f}_{G,h}(\mathbf{x}) \mathbf{m}_{G,h}(\mathbf{x}) \tag{4.71}$$

With:

$$\hat{f}_{G,h}(\mathbf{x}) = \frac{c_{g,d}}{Nh^d} \sum_{i=1}^{N} g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right) \tag{4.72}$$

Where we have introduced the notation $f_{A,b}(\mathbf{x})$ to refer to the kernel density estimate built as in equation (4.44) using multivariate kernel $A(\mathbf{x})$ with bandwidth $\mathbf{H}_i = b^2\mathbf{I}_d\ \forall\, i = 1, \ldots, N$.

And:

$$\mathbf{m}_{G,h}(\mathbf{x}) = \frac{\sum_{i=1}^{N} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{N} g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x} \tag{4.73}$$

Is defined as the *mean shift* vector or simply *mean shift*. We still keep the subindex notation previously introduced to reflect that the mean shift is evaluated using kernel $G(\mathbf{x})$ with bandwidth $\mathbf{H}_i = h^2\mathbf{I}_d\ \forall\, i = 1, \ldots, N$. The name comes from the fact that the *mean shift* is the difference between the weighted mean computed around location $\mathbf{x}$, with weights $g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)$, and the point $\mathbf{x}$ itself. If we use the Epanechnikov kernel, this is equivalent to saying that the *mean shift* is the difference between the mean of all points in a window of radius h around $\mathbf{x}$ and $\mathbf{x}$. Graphically, this is shown in figure 4.22

Equation (4.71) is extremely revealing. Indeed, we can rearrange the equation as:

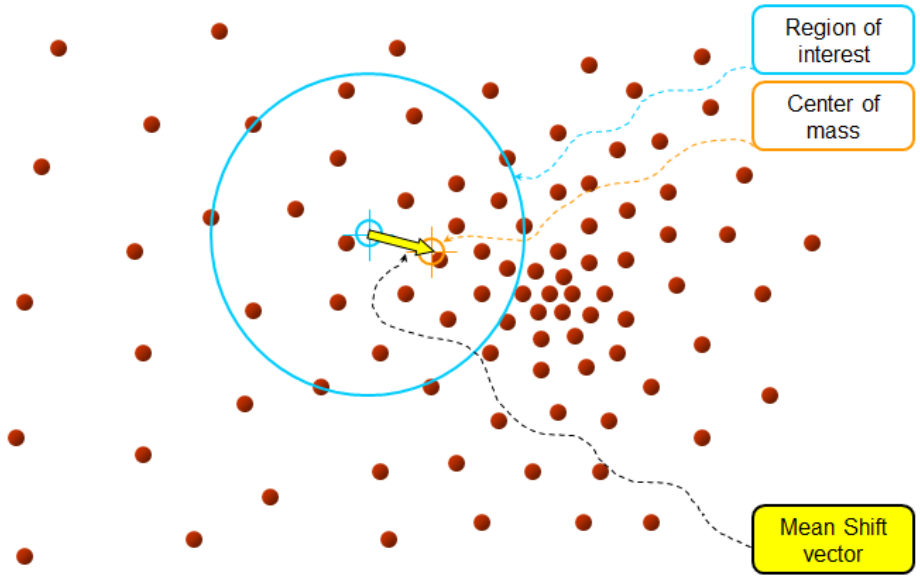


Figure 4.22: Mean shift vector as the difference between the weighted mean computed around location $\mathbf{x}$ *with weights* $g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)$*, and the point* $\mathbf{x}$ *itself. Epanechnikov kernel is used so that the weighted mean is simply the mean of all points in a* h*-ball around* $\mathbf{x}$*.*

$$\mathbf{m}_{G,h}(\mathbf{x}) = \frac{1}{2}\frac{c_{g,d}}{c_{k,d}}\frac{\nabla_{\mathbf{x}}\hat{f}_{K,h}(\mathbf{x})}{\hat{f}_{G,h}(\mathbf{x})} \tag{4.74}$$

Nonetheless, equation (4.74) is implying that the mean shift evaluated with kernel $G(\mathbf{x})$ is parallel to the gradient estimate obtained with its shadow kernel $K(\mathbf{x})$! Hence, the mean shift points toward the direction of maximum increase in the PDF.

This motivates to define an alternative iteration algorithm:

$$\mathbf{x} \leftarrow \mathbf{x} + \mathbf{m}_{G,h}(\mathbf{x}) \tag{4.75}$$

However, from (4.73), we have that this iteration is exactly the same as (4.69). In other words, the mean-shift algorithm can also be interpreted as a gradient ascent algorithm!

But that's not all. As an extra gift, the mean shift has self-adaptive properties, because it is normalized by the kernel density estimate computed with kernel $G(\mathbf{x})$ at point $\mathbf{x}$, $f_{G,h}(\mathbf{x})$. In other words, the step size is proportional to the inverse of the PDF estimate.

This means that in regions with a low probability density, where the real gradient of the PDF would be small, the step size becomes much bigger whereas in regions with a very high probability density, the step size decreases keeping the mean shift with a reasonable norm. But this is fully equivalent to having a gradient ascent procedure with an adaptive step size, big when we are far from the maximum and getting smaller as we approach convergence to avoid overshooting.

When we use a whole per-sample bandwidth matrix $\mathbf{H}_i$, the model is not so pretty but the iteration in (4.67) can be readily seen to keep all the relevant properties like the self-adaptive behavior of the close relation between the mean shift and the gradient. To be rigorous, if $\mathbf{H}_i = h_i\mathbf{I}$, the gradient and the mean-shift are still parallel. When we use the complete per-sample bandwidth matrix, the mean-shift and the gradient are no longer parallel, but they are still closely related and linked by the Mahalanobis metric induced with covariance matrix $\mathbf{H}^{-1}$. The self-adaptive behavior is exhibited in all the cases though.

Up to now, all the justifications we have given to show the usefulness of the iteration defined by (4.67) have been merely intuitive arguments. However, the next results will prove that this iteration will be meaningful under very mild assumptions.

Theorem 4. *The sequence* $\left\{f_{K,\mathbf{H}_i}(\mathbf{x}^{(t)})\right\}_{t=1}^{\infty}$ *induced by the iteration:*

$$\mathbf{x}^{(t+1)} \leftarrow \left(\sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right)\mathbf{H}_i^{-1}\right)^{-1} \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right)\mathbf{H}_i^{-1}\mathbf{x}_i$$

Is non-decreasing and convergent, provided that the kernel profile $k(x)$ *generating kernel* $K(\mathbf{x})$ *is a convex function.*

Proof. If the number of samples N in the data set is finite, the kernel density estimate will be a bounded function. Therefore, the sequence $\left\{f_{K,\mathbf{H}_i}(\mathbf{x}^{(t)})\right\}_{t=1}^{\infty}$ is bounded. Then, it suffices to show that it is also non-decreasing, i.e, that $S(t+1) = f_{K,\mathbf{H}_i}(\mathbf{x}^{(t+1)}) - f_{K,\mathbf{H}_i}(\mathbf{x}^{(t)}) \geq 0$. We can write:

$$S(t+1) = \frac{c_{k,d}}{N}\sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2}\left(k\left(\left\|\mathbf{x}^{(t+1)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right) - k\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right)\right)$$

Where we have introduced $\|\bullet\|_{\mathbf{H}_i}$ as the Mahalanobis norm with covariance matrix $\mathbf{H}_i^{-1}$.

Since the kernel profile $k(x)$ is a convex function, it satisfies:

$$k(x^{(t+1)}) - k(x^{(t)}) \geq \frac{dk(x^{(t)})}{dx}\left(x^{(t+1)} - x^{(t)}\right)$$

Introducing that into the previous equation and recalling the definition of kernel profile $g(x)$ as $g(x) = -\frac{dk(x)}{dx}$ we obtain:

$$S(t+1) \geq \frac{c_{k,d}}{N}\sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right)\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2 - \left\|\mathbf{x}^{(t+1)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right)$$

From the definition of Mahalanobis norm $\|\mathbf{x}\|_{\mathbf{H}_i}^2 = \mathbf{x}^T\mathbf{H}_i^{-1}\mathbf{x}$ we have that:

$$\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2 - \left\|\mathbf{x}^{(t+1)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2 = \left(\mathbf{x}^{(t)}\right)^T\mathbf{H}_i^{-1}\mathbf{x}^{(t)} - \left(\mathbf{x}^{(t+1)}\right)^T\mathbf{H}_i^{-1}\mathbf{x}^{(t+1)} - 2\left(\mathbf{x}^{(t)} - \mathbf{x}^{(t+1)}\right)^T\mathbf{H}_i^{-1}\mathbf{x}_i$$

Therefore:

$$
\begin{aligned}
S(t+1) \geq & \frac{c_{k,d}}{N} \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right) \left(\mathbf{x}^{(t)}\right)^T \mathbf{H}_i^{-1} \mathbf{x}^{(t)} - \\
& - \frac{c_{k,d}}{N} \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right) \left(\mathbf{x}^{(t+1)}\right)^T \mathbf{H}_i^{-1} \mathbf{x}^{(t+1)} - \\
& - \frac{2c_{k,d}}{N} \left(\mathbf{x}^{(t)} - \mathbf{x}^{(t+1)}\right)^T \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right) \mathbf{H}_i^{-1} \mathbf{x}_i
\end{aligned}
$$

But, because of the way the iteration is defined, we have that:

$$
\sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right) \mathbf{H}_i^{-1} \mathbf{x}_i = \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right) \mathbf{H}_i^{-1} \mathbf{x}^{(t+1)}
$$

Then we finally get:

$$
S(t+1) \geq \frac{c_{k,d}}{N} \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right) \left\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\right\|_{\mathbf{H}_i}^2
$$

And, since all terms in the previous summation are non-negative, then $S(t+1) \geq 0$, which concludes the proof. □

In [33] the authors prove a similar result as in theorem 4 for the particular case $\mathbf{H}_i = h^2 \mathbf{I}_d$. In that theorem, they also state that the sequence $\left\{\mathbf{x}^{(t)}\right\}_{t=1}^{\infty}$ is convergent. However, even though their article is certainly excellent, that statement and, more precisely, the particular proof provided for it, is incorrect. Indeed, their argument is based on the last equation obtained in the proof of theorem 4:

$$
S(t+1) \geq \frac{c_{k,d}}{N} \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right) \left\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\right\|_{\mathbf{H}_i}^2 \tag{4.76}
$$

Now, they introduced a summation operator on both sides of equation (4.76). Recalling our previous definition of $S(t+1)$ as $S(t+1) = f_{K,\mathbf{H}_i}(\mathbf{x}^{(t+1)}) - f_{K,\mathbf{H}_i}(\mathbf{x}^{(t)})$ the reader can easily see that the summation yields $\sum_{i=1}^{m} S(t+i) = f_{K,\mathbf{H}_i}(\mathbf{x}^{(t+m)}) - f_{K,\mathbf{H}_i}(\mathbf{x}^{(t)})$. Then, the equation becomes:

$$
\begin{aligned}
f_{K,\mathbf{H}_i}(\mathbf{x}^{(t+m)}) - f_{K,\mathbf{H}_i}(\mathbf{x}^{(t)}) \geq & \frac{c_{k,d}}{N}\sum_{i=1}^{N}|\mathbf{H}_i|^{-1/2}g\left(\left\|\mathbf{x}^{(t)}-\mathbf{x}_i\right\|^2_{\mathbf{H}_i}\right)\left\|\mathbf{x}^{(t+1)}-\mathbf{x}^{(t)}\right\|^2_{\mathbf{H}_i} + \\
& + \frac{c_{k,d}}{N}\sum_{i=1}^{N}|\mathbf{H}_i|^{-1/2}g\left(\left\|\mathbf{x}^{(t+1)}-\mathbf{x}_i\right\|^2_{\mathbf{H}_i}\right)\left\|\mathbf{x}^{(t+2)}-\mathbf{x}^{(t+1)}\right\|^2_{\mathbf{H}_i} + \\
& + \ldots + \\
& + \frac{c_{k,d}}{N}\sum_{i=1}^{N}|\mathbf{H}_i|^{-1/2}g\left(\left\|\mathbf{x}^{(t+m-1)}-\mathbf{x}_i\right\|^2_{\mathbf{H}_i}\right)\left\|\mathbf{x}^{(t+m)}-\mathbf{x}^{(t+m-1)}\right\|^2_{\mathbf{H}_i}
\end{aligned}
\tag{4.77}
$$

In the next step, the authors of [33] define:

$$
M = \min_{0\leq j\leq m-1}\frac{c_{k,d}}{N}\sum_{i=1}^{N}|\mathbf{H}_i|^{-1/2}g\left(\left\|\mathbf{x}^{(t+j)}-\mathbf{x}_i\right\|^2_{\mathbf{H}_i}\right) \tag{4.78}
$$

To be able to write:

$$
f_{K,\mathbf{H}_i}(\mathbf{x}^{(t+m)}) - f_{K,\mathbf{H}_i}(\mathbf{x}^{(t)}) \geq M\sum_{i=1}^{m}\left\|\mathbf{x}^{(t+i)}-\mathbf{x}^{(t+i-1)}\right\|^2_{\mathbf{H}_i} \tag{4.79}
$$

Up to this point, everything is correct. However, in the next step, they use the following (incorrect) inequality:

$$
\sum_{i=1}^{m}\left\|\mathbf{x}^{(t+i)}-\mathbf{x}^{(t+i-1)}\right\|^2_{\mathbf{H}_i} \geq \left\|\mathbf{x}^{(t+m)}-\mathbf{x}^{(t)}\right\|^2_{\mathbf{H}_i} \tag{4.80}
$$

As it was noted in [37], inequality (4.80) can be readily checked to be false even for extremely simple examples. For instance, a dummy example with only one dimension and $m = 2$ can be found with $\mathbf{x}^{(t)} = 1$, $\mathbf{x}^{(t+1)} = 2$ and $\mathbf{x}^{(t+2)} = 3$. For simplicity, let's also take $\mathbf{H}_i = 1$. Then we have that:

$$
\|3-2\|^2 + \|2-1\|^2 \ngeq \|3-1\|^2 \tag{4.81}
$$

Indeed, as it was pointed out by the authors of [37], the proof of theorem 1 in [33] is in blatant contradiction with the (correct) theorem 2 in the same article.

Luckily for us, apart from pointing out that minor mistake in [33], the main contribution of[37] was to find a correct proof of the empirically checked fact that the sequence $\left\{\mathbf{x}^{(t)}\right\}_{t=1}^{\infty}$ is convergent. They established the following theorem:

Theorem 5. *If the kernel profile $k(x)$ is convex and the number of singular points of the kernel density estimate $\hat{f}_{K,\mathbf{H}_i}(\mathbf{x})$ is finite on the set $\mathbb{S} = \left\{\mathbf{x} \mid \hat{f}_{K,\mathbf{H}_i}(\mathbf{x}) \geq \hat{f}_{K,\mathbf{H}_i}(\mathbf{x}^{(1)})\right\}$, the iterative*

sequence $\left\{\mathbf{x}^{(t)}\right\}_{t=1}^{\infty}$ *defined as:*

$$\mathbf{x}^{(t+1)} \leftarrow \left(\sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right) \mathbf{H}_i^{-1}\right)^{-1} \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right) \mathbf{H}_i^{-1}\mathbf{x}_i$$

Converges.

For the proof, we refer the reader to the original source of the theorem, [37]. As a technicality, in that source, the authors are a bit more rigorous and redefine the mean shift iteration to avoid trouble if the matrix inverse in the expression was ill-defined by stopping the iterative procedure in case that happens. However, in most practical cases, that is something that is not going to happen.

The implications of theorems 4 and 5 are extremely important. Using the relation between $\mathbf{x}^{(t+1)}$ and $\mathbf{x}^{(t)}$ implied by the mean-shift iterative procedure, we can rewrite equation (4.64) evaluated at $\mathbf{x} = \mathbf{x}^{(t)}$ as:

$$\hat{\nabla}_{\mathbf{x}} f(\mathbf{x}^{(t)}) = \frac{2c_{k,d}}{N} \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\left\|\mathbf{x}^{(t)} - \mathbf{x}_i\right\|_{\mathbf{H}_i}^2\right) \mathbf{H}_i^{-1}(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}) \tag{4.82}$$

Because the mean-shift sequence $\left\{\mathbf{x}^{(t)}\right\}_{t=1}^{\infty}$ is convergent with $\lim_{t\to\infty} \mathbf{x}^{(t)} = \mathbf{x}^{(c)}$, we have then that:

$$\nabla_{\mathbf{x}} \hat{f}_{K,\mathbf{H}_i}(\mathbf{x}^{(c)}) = \mathbf{0} \tag{4.83}$$

In other words, the mean-shift procedure is guaranteed to converge to a singular point of the kernel density estimate $\hat{f}_{K,\mathbf{H}_i}(\mathbf{x})$. Moreover, since the sequence of estimates $\left\{\hat{f}_{K,\mathbf{H}_i}(\mathbf{x}^{(t)})\right\}_{t=1}^{\infty}$ is non-decreasing, the *Capture Theorem* can be applied to conclude that once $\mathbf{x}^{(t)}$ gets sufficiently close to a local maximum $\mathbf{x}^*$ of $\hat{f}_{K,\mathbf{H}_i}(\mathbf{x})$, it will get attracted by $\mathbf{x}^*$ provided that $\mathbf{x}^*$ is the unique singular point within a small neighborhood centered in that maximum of $\hat{f}_{K,\mathbf{H}_i}(\mathbf{x})$.

In order to avoid false converge to saddle points or plateaus, it is usual to check that the singular point found, $\mathbf{x}^{(c)}$, indeed corresponds to a local maximum by perturbing $\mathbf{x}^{(c)}$ with a small random vector and running the mean-shift algorithm again until new convergence. If, with a reasonably small tolerance, the new point of convergence is the same than the original convergence point $\mathbf{x}^{(c)}$, then it was a local maximum. In figure 4.23 we see an example of convergence to a saddle point and how perturbing the convergence point and letting the algorithm run again can help detect this issue.

This motivates the following high-level algorithm for mean-shift based mode detection:

Algorithm 4.11 seems long and complicated but, in reality, it is quite simple. Given the data set and the kernel profile, together with the corresponding set of bandwidth matrices, which can be complete per-sample matrices or any of the simpler forms previously discussed, it tries to look for an estimate of the set of modes in the (unknown) PDF from which the data set is drawn.

Algorithm 4.11: Mean-shift based mode detection algorithm.

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \mid \mathbf{x}_i \in \mathbb{R}^d$
input : Kernel profile $k(x)$
input : Set of bandwidth matrices $\{\mathbf{H}_i\}_{i=1}^N$ with $\mathbf{H}_i \in \mathbb{R}^{d \times d}$
output: A set $\{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_{N_m}\} \mid \mathbf{m}_i \in \mathbb{R}^d$ with the estimates of the modes of the underlying PDF of data set $\mathbb{X}$

$p \leftarrow 0,\ \mathbb{M} \leftarrow \emptyset$;
$\mathbf{m}(\mathbf{x}) = \left(\sum_{i=1}^N |\mathbf{H}_i|^{-1/2} g\left(\|\mathbf{x} - \mathbf{x}_i\|^2_{\mathbf{H}_i}\right)\mathbf{H}_i^{-1}\right)^{-1} \sum_{i=1}^N |\mathbf{H}_i|^{-1/2} g\left(\|\mathbf{x} - \mathbf{x}_i\|^2_{\mathbf{H}_i}\right)\mathbf{H}_i^{-1}\mathbf{x}_i$;
Define $\mathbf{x}_{\min} \in \mathbb{R}^d$ as $\mathbf{x}^j_{\min} = \min\limits_{1 \le i \le N} \mathbf{x}^j_i$ and $\mathbf{x}_{\max} \in \mathbb{R}^d$ as $\mathbf{x}^j_{\max} = \max\limits_{1 \le i \le N} \mathbf{x}^j_i$;
Sample the rectangular box defined by $\mathbb{R} = \left\{\mathbf{y} \mid \mathbf{x}^j_{\min} \le \mathbf{y}^j \le \mathbf{x}^j_{\max}\ \forall\, j = 1, \ldots, d\right\}$ to obtain a set of initial points $\mathbb{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M\} \mid \mathbf{y}_i \in \mathbb{R}^d$ where the number of probe points M is a resolution dependent parameter ;
for $j \leftarrow 1$ **to** M **do**
    $t \leftarrow 0,\ \mathbf{x}^{(1)} \leftarrow \mathbf{y}_j$;
    while *convergence criterion not satisfied* **do**
        $t \leftarrow t + 1,\ \mathbf{x}^{(t+1)} \leftarrow \mathbf{m}\left(\mathbf{x}^{(t)}\right)$;
    end
    $\mathbf{x}^{(c)} \leftarrow \mathbf{x}^{(t)}$;
    Perturb $\mathbf{x}^{(c)}$ as $\mathbf{x}_\epsilon = \mathbf{x}^{(c)} + \left\|\mathbf{x}^{(c)}\right\| \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon}$ being a random vector drawn from a d-variate Gaussian distribution with small variance, in the order of 0.01 to 0.1 ;
    $t \leftarrow 0,\ \mathbf{x}^{(1)} \leftarrow \mathbf{x}_\epsilon$;
    while *convergence criterion not satisfied* **do**
        $t \leftarrow t + 1,\ \mathbf{x}^{(t+1)} \leftarrow \mathbf{m}\left(\mathbf{x}^{(t)}\right)$;
    end
    if $\left\|\mathbf{x}^{(t)} - \mathbf{x}^{(c)}\right\| < \text{tol}$ **then**
        $\mathbf{x}^{(c)}_p \leftarrow \mathbf{x}^{(c)}$;
        $p \leftarrow p + 1,\ \mathbb{M} \leftarrow \mathbb{M} \cup \mathbf{x}^{(c)}_p$;
    end
end
Partition $\mathbb{M}$ in subsets $\mathbb{M}_i = \left\{\mathbf{x}^{(c)}_j \mid \exists\, \mathbf{x}^{(c)}_k \in \mathbb{M}_i \mid \left\|\mathbf{x}^{(c)}_j - \mathbf{x}^{(c)}_k\right\| \le \text{tol}_h\right\}$. That is, a ball of radius tol_h centered at any point of $\mathbb{M}_i$ must contain at least another point of $\mathbb{M}_i$;
Let N_m be the number of subsets $\mathbb{M}_i$ found ;
for $j \leftarrow 1$ **to** N_m **do**
    $\mathbf{m}_j \leftarrow \frac{1}{|\mathbb{M}_i|} \sum_{\left\{j \mid \mathbf{x}^{(c)}_j \in \mathbb{M}_i\right\}} \mathbf{x}^{(c)}_j$
end

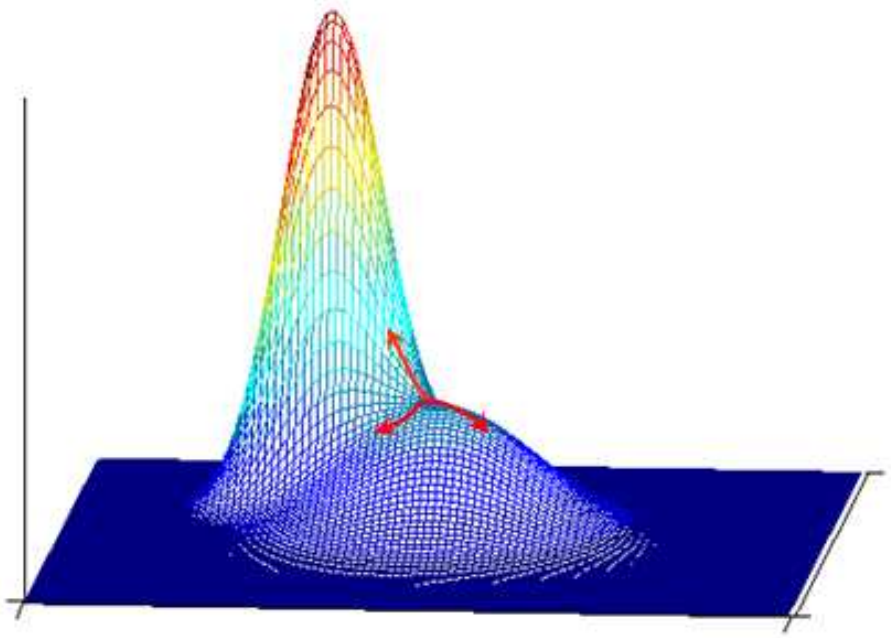


Figure 4.23: Example of convergence to a saddle point and the usage of random perturbation to detect this problem.

In order to do that, the first step is to generate a grid in the hyper-box $\mathbb{R}$ containing the original data set. Then, we run a mean-shift procedure for each of the M points $\mathbf{y}_j$ we have in the grid setting $\mathbf{y}_j$ as the initial point in the iteration. Note that the bigger the number of sampled points M, the bigger the resolution of the algorithm, at the expense of increased computational complexity.

Once we detect convergence of the mean-shift iterations to a point $\mathbf{x}^{(c)}$, we perturb it by adding a small random vector $\left\|\mathbf{x}^{(c)}\right\| \boldsymbol{\epsilon}$ and execute mean-shift again using the perturbed convergence point as the new seed for the iteration. If the new convergence point is the same (within a certain tolerance) than the original convergence point $\mathbf{x}^{(c)}$, then we accept $\mathbf{x}^{(c)}$ as a valid mode candidate $\mathbf{x}_i^{(c)}$ and store it in the set of mode candidates $\mathbb{M}$. On the other hand, if perturbing the convergence point $\mathbf{x}^{(c)}$ and letting it converge again yields a different result, $\mathbf{x}^{(c)}$ was not a local maximum but a saddle point and we discard it.

Finally, it is necessary to prune the set of mode candidates $\mathbb{M}$. For that, any subset of mode candidates which are sufficiently close to each other are considered to be essentially the same mode, being any discrepancy due to the limitations of the implicit gradient ascent procedure and/or numerical issues. Therefore, we find subsets of $\mathbb{M}$ such that for any given point in the subset, there is at least another point in the subset at a distance less than a prefixed mode-seeking tolerance tol_h.

More elaborated mode pruning procedures can be found in the literature. For instance, [38] observes that, between any two modes, a valley should occur in the PDF estimate. Therefore,

for any pair of mode candidates $\mathbf{x}_i^{(c)}$, $\mathbf{x}_j^{(c)}$, the kernel density estimate is evaluated in the line joining $\mathbf{x}_j^{(c)}$ with $\mathbf{x}_k^{(c)}$. If the ratio between the min $\left(\hat{f}_{K,\mathbf{H}_i}(\mathbf{x}_j^{(c)}), \hat{f}_{K,\mathbf{H}_i}(\mathbf{x}_k^{(c)})\right)$, that is, the lowest value between the two peaks of the PDF estimate we are looking at and the minimum value of the PDF estimate along the line joining the modes (valley value) is below a prefixed threshold T, then the mode $\mathbf{x}_j^{(c)}$ or $\mathbf{x}_k^{(c)}$ with the lowest density estimate is removed from the set of mode candidates.

In the next chapter, we will discuss the mode pruning procedure we developed in this project to tackle the special needs of base-station clustering for coordinated transmission in a mobile communications scenario with MIMO transceivers.

4.5.4 Mean-shift based clustering

The relation between mode-seeking and clustering is straightforward and was pointed out in practically all the main articles on mean shift such as [31], [32] or [33].

Since each mode in $\{\mathbf{m}_i\}_{i=1}^{N_m}$ acts as an attractor in $\mathbb{R}^d$ under the implicit gradient ascent procedure defined by the mean shift iterations, we can define the *basin of attraction* of each mode $\mathbf{m}_i$ as the set of points in $\mathbb{R}^d$ which converge to $\mathbf{m}_i$ under the mean-shift iterations. It is very easy to see that the partition of $\mathbb{R}^d$ generated by the distinct *basins of attraction* can be used as a very reasonable way to partition the original data set. A graphical illustration of a simple data set and its *basins of attraction* is given in figure 4.24.

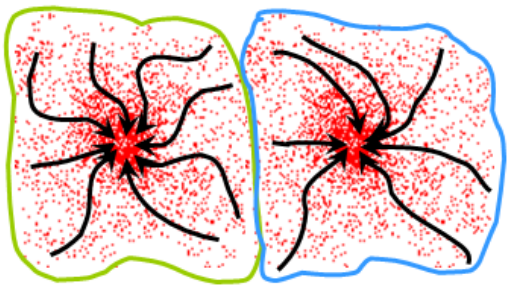


Figure 4.24: Attraction basins in a simple data set inducing a natural choice for clusters.

That observation inspires the following clustering algorithm:

The reader can see that the clustering algorithm 4.12 is extremely similar to the mode-seeking algorithm based on meanshift written in 4.11.

We have the same inputs: the data set of points $\mathbb{X}$, the kernel profile $k(x)$ and the set of bandwidth matrices $\{\mathbf{H}_i\}_{i=1}^{N}$. Just like before, we run several mean-shift iterations in parallel with different seeds. However, in this case, the initial points of the iterations are now the points $\mathbf{x}_i$ in the data set $\mathbb{X}$. In other words, no sampling in a grid containing the data set is performed as we did in the mode-seeking algorithm.

Then, each point $\mathbf{x}_i$ is used as the initial point in a mean-shift iteration. The limit point of the iteration, $\mathbf{x}^{(c)}$, is tested by perturbing it and letting it converge again, just as with algorithm

Algorithm 4.12: Mean-shift based clustering algorithm.

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \mid \mathbf{x}_i \in \mathbb{R}^d$
input : Kernel profile $k(x)$
input : Set of bandwidth matrices $\{\mathbf{H}_i\}_{i=1}^N$ with $\mathbf{H}_i \in \mathbb{R}^{d\times d}$
output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_{N_m}\}$ of $\mathbb{X}$

$p \leftarrow 0,\ \mathbb{M} \leftarrow \emptyset$;
$\mathbf{m}(\mathbf{x}) = \left(\sum_{i=1}^N |\mathbf{H}_i|^{-1/2} g\left(\|\mathbf{x}-\mathbf{x}_i\|^2_{\mathbf{H}_i}\right)\mathbf{H}_i^{-1}\right)^{-1} \sum_{i=1}^N |\mathbf{H}_i|^{-1/2} g\left(\|\mathbf{x}-\mathbf{x}_i\|^2_{\mathbf{H}_i}\right)\mathbf{H}_i^{-1}\mathbf{x}_i$;
for $j \leftarrow 1$ **to** N **do**
    $t \leftarrow 0,\ \mathbf{x}^{(1)} \leftarrow \mathbf{x}_j$;
    while *convergence criterion not satisfied* **do**
        $t \leftarrow t+1,\ \mathbf{x}^{(t+1)} \leftarrow \mathbf{m}\left(\mathbf{x}^{(t)}\right)$;
    end
    $\mathbf{x}^{(c)} \leftarrow \mathbf{x}^{(t)}$;
    Perturb $\mathbf{x}^{(c)}$ as $\mathbf{x}_\epsilon = \mathbf{x}^{(c)} + \left\|\mathbf{x}^{(c)}\right\| \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon}$ being a random vector drawn from a d-variate Gaussian distribution with small variance, in the order of 0.01 to 0.1 ;
    $t \leftarrow 0,\ \mathbf{x}^{(1)} \leftarrow \mathbf{x}_\epsilon$;
    while *convergence criterion not satisfied* **do**
        $t \leftarrow t+1,\ \mathbf{x}^{(t+1)} \leftarrow \mathbf{m}\left(\mathbf{x}^{(t)}\right)$;
    end
    if $\left\|\mathbf{x}^{(t)} - \mathbf{x}^{(c)}\right\| < \text{tol}$ **then**
        $\mathbf{x}_p^{(c)} \leftarrow \mathbf{x}^{(c)}$;
        $p \leftarrow p+1,\ \mathbb{M} \leftarrow \mathbb{M} \cup \mathbf{x}_p^{(c)}$;
        Associate point $\mathbf{x}_j$ with mode candidate $\mathbf{x}_p^{(c)}$: $C(j) = p$;
    else
        Discard point $\mathbf{x}^{(c)}$ since it corresponds to a saddle point ;
        Annotate that point $\mathbf{x}_j$ has no associated mode candidate for now: $C(j) = -1$;
    end
end
Partition $\mathbb{M}$ in subsets $\mathbb{M}_i = \left\{\mathbf{x}_j^{(c)} \mid \exists\, \mathbf{x}_k^{(c)} \in \mathbb{M}_i \mid \left\|\mathbf{x}_j^{(c)} - \mathbf{x}_k^{(c)}\right\| \leq \text{tol}_h\right\}$. That is, a ball of radius tol_h centered at any point of $\mathbb{M}_i$ must contain at least another point of $\mathbb{M}_i$;
Each of the subset $\mathbb{M}_1, \mathbb{M}_2, \ldots, \mathbb{M}_{N_m}$ defines a distinct cluster ;
for $j \leftarrow 1$ **to** N_m **do**
    $\mathbf{m}_j \leftarrow \frac{1}{|\mathbb{M}_i|} \sum_{\left\{j \mid \mathbf{x}_j^{(c)} \in \mathbb{M}_i\right\}} \mathbf{x}_j^{(c)}$
end
$\mathbb{S}_k \leftarrow \left\{\mathbf{x}_i \mid \mathbf{x}_{C(i)}^{(c)} \in \mathbb{M}_k\right\} \cup \left\{\mathbf{x}_i \mid C(i) = -1, k = \min_{1\leq l\leq N_m} \|\mathbf{x}_i - \mathbf{m}_l\|\right\}$;

4.11. If $\mathbf{x}^{(c)}$ passes the test, it will be accepted as a mode candidate $\mathbf{x}_p^{(c)}$ and the fate of data point $\mathbf{x}_i$ will be forever linked to that of mode candidate $\mathbf{x}_p^{(c)}$. We annotate that by letting $C(i) = p$. However, if $\mathbf{x}^{(c)}$ fails the test, its potential mate $\mathbf{x}_i$ is left, for now, orphan. We then associate $C(i) = -1$.

Once all mode candidates have been found and stored in $\mathbb{M}$, and the relations between data points and mode candidates have been annotated in C, we use the same pruning procedure as in 4.11 to collapse nearby modes into a reduced set of estimated modes $\{\mathbf{m}_i\}_{i=1}^{N_m}$.

Then, the observation $\mathbf{x}_j$ will be assigned to cluster k if its associated mode candidate $\mathbf{x}_{C(j)}^{(c)}$ belongs to subset $\mathbb{M}_k$, that is, if its associated mode candidate was fused into mode $\mathbf{m}_k$. The orphan points $\mathbf{x}_j$ without an associated mode candidate, those with $C(j) = -1$, are assigned to the cluster corresponding to the closest mode $\mathbf{m}_k$ within the set of estimated modes $\{\mathbf{m}_i\}_{i=1}^{N_m}$.

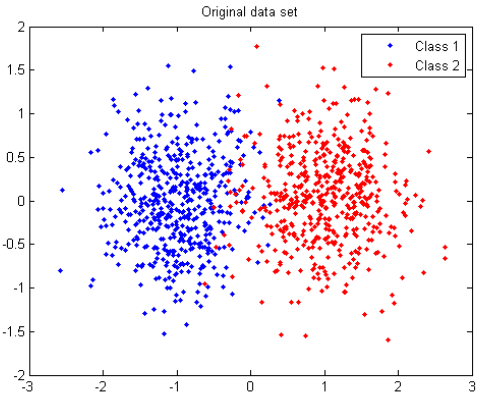
An example with a toy dataset is shown in figure 4.25. 1000 bi-dimensional points have been drawn from a bimodal bivariate gaussian distribution with modes in $\mathbf{m}_1 = (1, 0)$ and $\mathbf{m}_2 = (-1, 0)$.

Both components in the mixture are radially symmetric with $\mathrm{Tr}(\mathbf{\Sigma}) = 1.28$. In the upper row, we show the original data set and the clusters obtained with algorithm 4.12. As we can see, the algorithm only fails with points which are closer to the other mode. The most interesting point in the example can be found in the figures showing the trajectories defined by the mean-shift iterations: 4.25(e) and 4.25(f). We know that, if we use an scalar bandwidth matrix $\mathbf{H} = h^2\mathbf{I}$, as we did in the example, then the mean-shift vector is in theory parallel to the gradient. Indeed, we can clearly see that the trajectories are perpendicular to the level curves of the estimated PDF, showing empirically that the theory reflect reality.

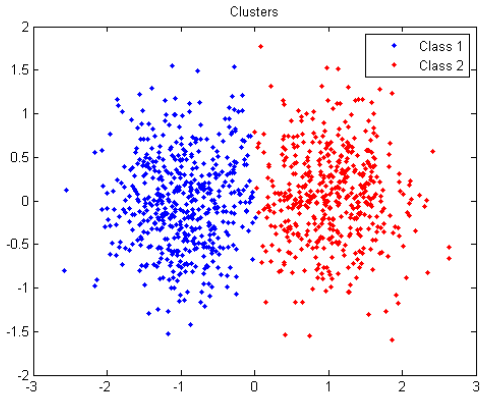
Extensions and alternatives

Apart from using more elaborated mode pruning procedures as the ones we briefly pointed out when discussing algorithm 4.11, it is possible to create synergies between mean-shift based clustering and other well-known clustering algorithms. For instance, in [39], the authors propose what they denote as a mean-shift spectral clustering algorithm. Actually, what they really propose is nothing more, but nothing less, that a mean-shift clustering algorithm where modes are pruned by collapsing them using spectral clustering. In other words, it is a two-step clustering method in which nearby modes are clustered together using spectral clustering. The most interesting point of their work is the discussion on which similarity function should be used in the spectral clustering algorithm to obtain something consistent with the inherent kernel density estimation procedure done by mean-shift. In [38], they propose a mixture between mean-shift mode seeking and K-NN classification, so that clusters are delineated by majority-vote with the K-nearest modes of each data point. Also, later in this project, we develop another example of an improved mean-shift algorithm by designing an evolutionary mean-shift clustering algorithm for base-station grouping.

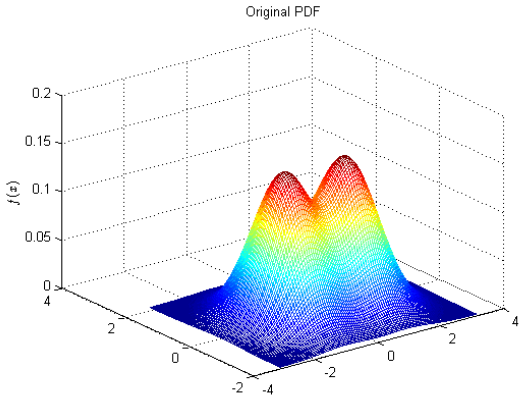
Another interesting point to discuss, even though it may seem a bit pointless at first, is the



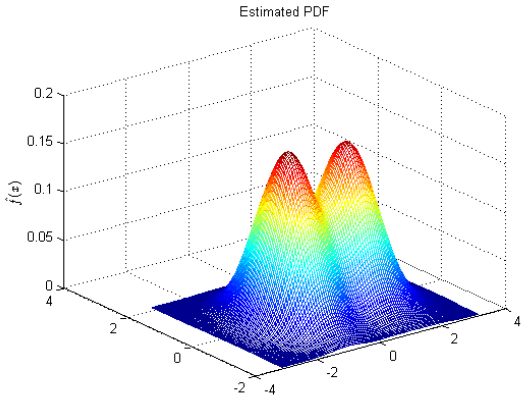
(a) Original data set



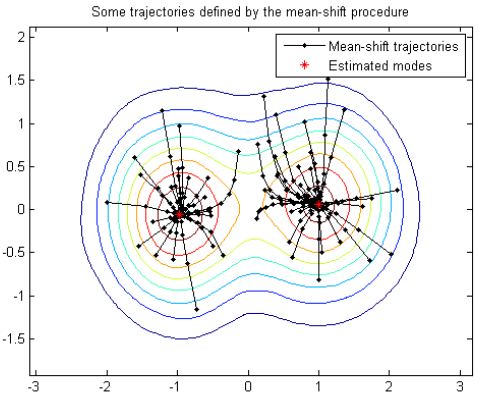
(b) Clusters obtained with algorithm 4.12



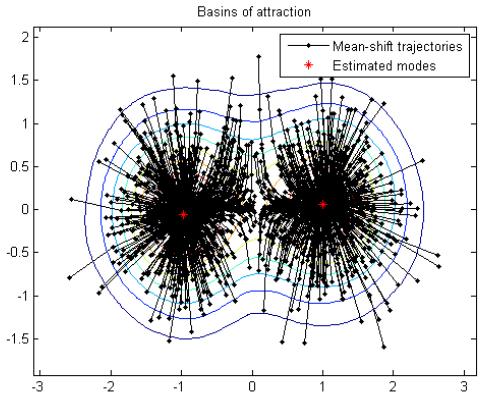
(c) Original PDF



(d) Estimated PDF with Normal kernel and $\mathbf{H} = \mathbf{I}_2$



(e) Some mean-shift trajectories superimposed in a contour plot of the estimated PDF.



(f) Attraction basins (all trajectories shown at the same time)

Figure 4.25: Mean-shift clustering example.

fact that it is possible to use a data set $\mathbb{Y}$ as a training set for the kernel density estimator and the different data set $\mathbb{X} \neq \mathbb{Y}$ as a set of initial locations for the mean shift iterations. Because the clusters are defined according to the convergence of each initial seed to one of the attraction basins, it is data set $\mathbb{X}$ the one which is clustered. However, $\mathbb{Y}$ is responsible for creating the attraction basins which induce the partition.

We include that modification in algorithm 4.13.

As we can see, the differences between algorithm 4.13 and algorithm 4.12 are really small: basically, we have now a new data set $\mathbb{Y}$ which is employed to evaluate the mean-shift update function is line 2. The rest is pretty much unchanged.

This is probably suboptimal in most typical clustering scenarios and has been treated in the literature only as a way of reducing the complexity (see, for instance, [33] or [38]). Nonetheless, in this project, we will exploit this extra flexibility to improve the performance of mean-shift for clustering of base-stations in mobile communications by adapting the mean-shift procedure to the context of the application. We will discuss that in detail in the next chapter.

4.6 Clustering with Evolutionary Computation

An evolutionary algorithm can be regarded as a search heuristic inspired by the process of natural evolution. They are one of the best ways to solve optimization problems for which little a priori knowledge is available. Besides, they are completely general, so that we can be apply them to any problem as long as we define a customized fitness function which, given a candidate solution, evaluates "how good" such a solution is. Therefore, evolutionary methods are especially useful in problems with a large search space with many local maxima and minima. Their performance is usually excellent, outperforming most of the alternative algorithms currently available. However, the fundamental weakness of evolutionary algorithms is that, due to the extreme computational burden involved, it is unfeasible to use them in many real world applications.

Strictly speaking, evolutionary computation methods encompass several classes of methods and algorithms. For instance, sticking to precise definitions, genetic algorithms are those which employ binary representations of the solutions, with mutation and crossover as genetic operators; evolutionary search uses real-valued arrays instead and relays mostly on mutation; and genetic programming exploits tree-like representations and defines genetic operators suitable for such an encoding of solutions to allow programs to evolve. And this is only to name a few.

Nonetheless, after the popularization of evolutionary computation, so many algorithms that blend concepts from several distinct classes of evolutionary methods have been proposed that, for the remainder of this section, we will continuously refer to any algorithm which tries to mimic natural evolution to find the solution of a problem as a *genetic algorithm*. We will do so even if the representation of the solutions is not a bit string or the genetic operators are different that mutation and crossover.

Genetic algorithms are named like that because the heuristic in which they are based is

Algorithm 4.13: Generalized mean-shift based clustering algorithm.

input : A data set $\mathbb{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\} \mid \mathbf{y}_i \in \mathbb{R}^d$ to be used for kernel density estimation

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \mid \mathbf{x}_i \in \mathbb{R}^d$ to be used as seeds in the gradient ascent procedure

input : Kernel profile $k(x)$

input : Set of bandwidth matrices $\{\mathbf{H}_i\}_{i=1}^N$ with $\mathbf{H}_i \in \mathbb{R}^{d \times d}$

output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_{N_m}\}$ of $\mathbb{X}$

$p \leftarrow 0$, $\mathbb{M} \leftarrow \emptyset$;

$\mathbf{m}(\mathbf{x}) = \left(\sum_{i=1}^N |\mathbf{H}_i|^{-1/2} g\left(\|\mathbf{x} - \mathbf{y}_i\|^2_{\mathbf{H}_i}\right) \mathbf{H}_i^{-1}\right)^{-1} \sum_{i=1}^N |\mathbf{H}_i|^{-1/2} g\left(\|\mathbf{x} - \mathbf{y}_i\|^2_{\mathbf{H}_i}\right) \mathbf{H}_i^{-1} \mathbf{y}_i$;

for $j \leftarrow 1$ **to** M **do**

  $t \leftarrow 0$, $\mathbf{x}^{(1)} \leftarrow \mathbf{x}_j$;

  while *convergence criterion not satisfied* **do**

    $t \leftarrow t + 1$, $\mathbf{x}^{(t+1)} \leftarrow \mathbf{m}\left(\mathbf{x}^{(t)}\right)$;

  end

  $\mathbf{x}^{(c)} \leftarrow \mathbf{x}^{(t)}$;

  Perturb $\mathbf{x}^{(c)}$ as $\mathbf{x}_\epsilon = \mathbf{x}^{(c)} + \left\|\mathbf{x}^{(c)}\right\| \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon}$ being a random vector drawn from a d-variate Gaussian distribution with small variance, in the order of 0.01 to 0.1 ;

  $t \leftarrow 0$, $\mathbf{x}^{(1)} \leftarrow \mathbf{x}_\epsilon$;

  while *convergence criterion not satisfied* **do**

    $t \leftarrow t + 1$, $\mathbf{x}^{(t+1)} \leftarrow \mathbf{m}\left(\mathbf{x}^{(t)}\right)$;

  end

  if $\left\|\mathbf{x}^{(t)} - \mathbf{x}^{(c)}\right\| < \text{tol}$ **then**

    $\mathbf{x}_p^{(c)} \leftarrow \mathbf{x}^{(c)}$;

    $p \leftarrow p + 1$, $\mathbb{M} \leftarrow \mathbb{M} \cup \mathbf{x}_p^{(c)}$;

    Associate point $\mathbf{x}_j$ with mode candidate $\mathbf{x}_p^{(c)}$: $C(j) = p$;

  else

    Discard point $\mathbf{x}^{(c)}$ since it corresponds to a saddle point ;

    Annotate that point $\mathbf{x}_j$ has no associated mode candidate for now: $C(j) = -1$;

  end

end

Partition $\mathbb{M}$ in subsets $\mathbb{M}_i = \left\{\mathbf{x}_j^{(c)} \mid \exists\, \mathbf{x}_k^{(c)} \in \mathbb{M}_i \mid \left\|\mathbf{x}_j^{(c)} - \mathbf{x}_k^{(c)}\right\| \leq \text{tol}_h\right\}$. That is, a ball of radius tol_h centered at any point of $\mathbb{M}_i$ must contain at least another point of $\mathbb{M}_i$;

Each of the subset $\mathbb{M}_1, \mathbb{M}_2, \ldots, \mathbb{M}_{N_m}$ defines a distinct cluster ;

for $j \leftarrow 1$ **to** N_m **do**

  $\mathbf{m}_j \leftarrow \frac{1}{|\mathbb{M}_i|} \sum_{\left\{j \mid \mathbf{x}_j^{(c)} \in \mathbb{M}_i\right\}} \mathbf{x}_j^{(c)}$

end

$\mathbb{S}_k \leftarrow \left\{\mathbf{x}_i \mid \mathbf{x}_{C(i)}^{(c)} \in \mathbb{M}_k\right\} \cup \left\{\mathbf{x}_i \mid C(i) = -1, k = \min_{1 \leq l \leq N_m} \|\mathbf{x}_i - \mathbf{m}_l\|\right\}$;

strongly inspired by genetics. As a reference, we hint at some connections between biochemistry, genetics and their computational counterparts in appendix B.

4.6.1 Introduction to genetic algorithms

Let us suppose that we face a certain optimization problem, in which we have to find, within a certain search space containing all feasible solutions to the problem, the one which works better.

The most straightforward (and stupid) way to tackle this problem is to randomly pick candidate solutions from the search space and test them one by one. After a certain number of possible solutions have been tested, we retain the one which performed better and that would be the algorithm's output. However, such a primitive, brute force attempt can only work in extremely simple problems with a very reduced search space. Even though, according to the *infinite monkey theorem*, that algorithm would eventually reach the global optimum, the expected time to achieve that would diverge.

Genetic algorithms are essentially an heuristically improved version of that brute force approach, in which the candidate solutions which are picked to be evaluated are generated employing natural evolution.

At first, we randomly generate a set of m candidate solutions from the search space. Following the parallelism with biology, each candidate solution is said to be an *individual* and the whole set the *population*, being m *the population size*. Each individual in the population is then tested with the objective function, which in the context of genetic algorithms is denoted as *fitness function*. Therefore, individuals which represent good solutions are said to be fitter that those which represent solutions with poor performance.

The next step in the process is to apply Darwin's survival of the fittest principle: individuals from the population are randomly chosen to fill a mating pool in such a way that fitter individuals have a greater chance of generating offspring. The stochastic nature of the selection process is essential to correctly mimic natural evolution: being fitter implies a greater chance of reproducing but it does not guarantee it. Sometimes weak individuals can be charming too! Even though this may seem to be suboptimal, both in evolutionary computing and in biological evolution, stochastic natural selection benefits the whole optimization process. From an intuitive point of view, an individual which is overall weak, may have still some beneficial traits which would be lost if it did not get to reproduce at all. If the natural selection process was deterministic, fitter individuals would cannibalize the population, extremely reducing the genetic variability. In terms of optimization, this implies premature convergence to a local optimum.

Once e individuals have been chosen to reproduce, we group them in pairs and let them have offspring with probability p_c. Inspired in biological meiosis, the descendants are generated as a mixture or crossing-over of the two parents. Usually, two children are generated to replace the two parents in the next generation. In case they randomly happen to not have any offspring, the most typical choice is to copy the parents onto the next generation to keep the population size constant. However this is up to the engineer.

Finally, once the individuals for the next generation have been created through crossing of the previous generation's individuals, we apply a mutation process to each new individual. In this process, the most usual choice is to randomly introduce, with probability p_m, small changes in the characteristics of each individual, inspired by the random errors in the DNA replication process which originate biological mutations.

Algorithm 4.14: Basic outline of a generic genetic algorithm

input : Fitness evaluation function $f(x)$
input : Population size m
input : Mating pool size e
input : Crossover probability p_c and mutation probability p_m
output: An individual x^* which attempts (with no guarantees) to optimize the fitness function $f(x)$

```
Initialize m individuals randomly ;
Evaluate the fitness f(x) of each of the m individuals x in the population ;
while convergence criterion not satisfied do
    Apply selection to fill a mating pool of size e with individuals from the population ;
    Shuffle the mating pool and divide it in e/2 pairs of individuals ;
    Apply crossover to each pair with probability p_c to obtain two new individuals.
    Otherwise, copy the original pair as if they were the children ;
    Apply mutation to each individual in the offspring with probability p_m ;
    Replace the current generation with the offspring ;
    Evaluate the fitness f(x) of each of the new individuals x in the population ;
end
Output the fittest individual of the last generation ;
```

Algorithm 4.14 is simple the basic skeleton used to build genetic algorithms. During the remaining of this section, we will discuss how to tweak that basic skeleton to best fit our application's needs. For instance, crossover or mutation operators could be removed or changed by other genetic operators, or certain ideas like keeping the fittest individuals of one generation onto the next, the so-called *elitism* strategy, could be included into 4.14.

Genetic algorithms work exactly because of the same reasons natural evolution works.

Crossing allows two solutions to interchange their characteristics. The offspring of two solutions may happen to inherit the bad traits of its parents, and have a smaller fitness than them. On the other hand, it could inherit the good traits of each parent, so that the new individual is an improved version of both original candidate solutions. The mutation process allows to explore new parts of the search space and to escape premature convergence to local optimums at the expense of often reducing the fitness. If we have a sufficiently large population, the stochastic nature of both phenomena will imply that even though some new individuals will be less fit

than its parents, we will also get some which are fitter. Since fitter individuals get to have more children, the average fitness of the population usually increases with each generation, and failed individuals created through crossing and mutation are promptly eliminated from the solution pool.

In terms of machine learning, crossing and mutation allows *exploration* of the search space whereas selection allows for *exploitation*.

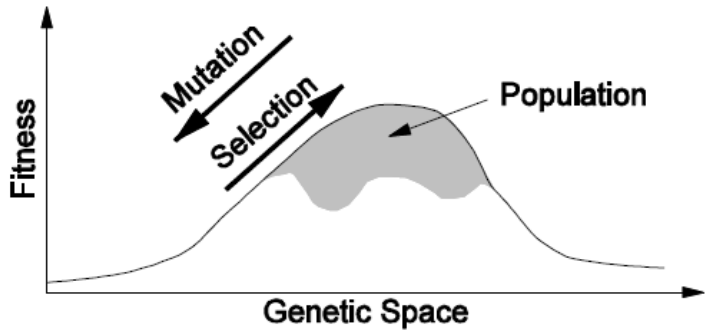


Figure 4.26: Selection is the key to exploitation in genetic algorithms. However, too much selection pressure can lead to premature convergence.

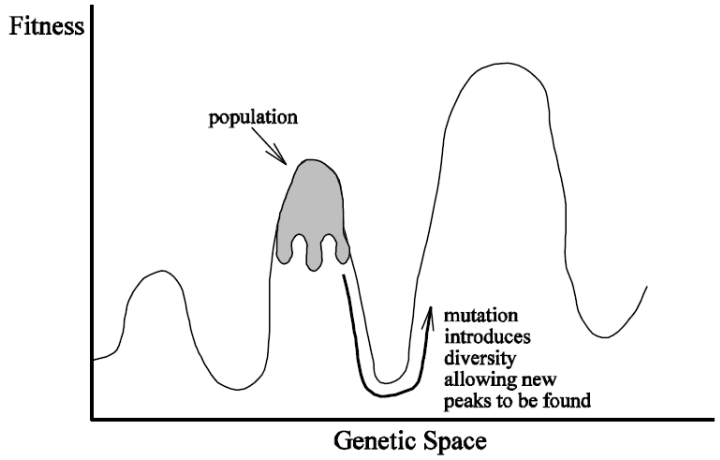


Figure 4.27: Mutation allows exploration of the search space, avoiding premature convergence to local optimum points.

4.6.1.1 Genetic representation of individuals

Choosing an appropriate representation of the candidate solutions is what ultimately determines the success of complete failure of a particular implementation of an evolutionary algorithm.

Following the biological analogy, the data structure which uniquely characterizes an individual (candidate solution) is called a *chromosome*. Each of the fields in the data structure is said to be a *gene*, and the set of values each *gene* can take are denoted as *alleles*.

The genetic operators crossing and mutation will be implemented in a different manner depending on the particular structure chosen as chromosome. Therefore, finding an appropriate representation of the candidate solutions is also the first decision to be made when designing a genetic algorithm.

Binary encoding

The first genetic algorithms, nowadays known as Simple Genetic Algorithm (SGA), was proposed by John Henry Holland in [40], published in 1975. He was the first one to think of applying natural selection to machine intelligence. In his original algorithm, Holland used a extremely simple representation of solutions: a bit string of fixed length. For instance, chromosome a may be $c_a = (1, 1, 0, 0, 1, 0, 1)$ and chromosome b $c_b = (1, 0, 0, 1, 1, 1, 0)$.

For instance, this representation can be used straightforwardly to solve the Knapsack problem. Let us suppose that the have a single bag with fixed capacity C and a set of objects $\{o_i\}_{i=1}^{N}$ which we would like to carry with us. Each object o_i is characterized by its value, v_i, and its size, c_i. Our problem is to pick, from the set of objects, those which maximize the total value of the knapsack's content without exceeding its capacity.

A very simple way of encoding a solution is using a bit string of length N. A chromosome whose i-th gene has allele 1 implies that the i-th object is carried in the knapsack. Similarly, allele 0 would code that the object is discarded.

Other problems where binary encoding can be used are those with numeric variables such as integers or real numbers. By using any machine representation, like 2's complement of IEEE floating point numbers, it is possible to represent that kind of variables with binary chromosomes. However, there are better encodings available for that task.

A binary chromosome can encode lots of solutions with a small number of genes, since the number of individuals which can be encoded grows exponentially with the length of the chromosome. However, it is easy to see that, for many problems, this encoding is quite unnatural and it may be hard to find a mapping from the search space to the set of binary strings of length N. The main advantage of Holland's original scheme is its great simplicity. As we will see later, genetic operators for binary chromosomes are really easy to implement, reducing the computational burden of the algorithm to basically fitness evaluation.

Permutation encoding

Another important type of representation is permutation encoding. As the name suggests, it is engineered for ordering problems, where we have a set of N elements and we want to find the optimal way to arrange the elements in a sequence. Note that, since we have $N!$ possibilities, this type of problems scales terribly bad with the number of elements N.

Candidate solutions of an ordering problem can be represented by a permutation of N elements, $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ with $\pi_i \in \{1, 2, \ldots, N\}$ and $\pi_i \neq \pi_j \; \forall \; i \neq j$. Precisely, $\pi_i = k$ means that the i-th element is to be placed in the k-th position.

Therefore, the most natural chromosomal representation for ordering problems of N-elements is with a N-dimensional vector of natural numbers in the set $\{1, 2, \ldots, N\}$ with the additional constraint of not allowing any repetitions in the array. For instance, valid chromosomes for ordering 5 objects could be $c_a = (3, 2, 5, 4, 1)$ or $c_b = (1, 5, 4, 2, 3)$.

Permutation encoding use genetic operators specifically designed to ensure that the next generation still represents a valid solution, that is, that new chromosomes still represent a permutation.

A typical problem for which permutation encoding can be applied is the Travelling Salesman Problem (TSP). Consider a poor salesman which has to visit a set of N distinct cities. Since he does not have much money for petrol, he wants to find the order in which he should visit the N cities to minimize the total distance traveled. For small N, this may be easy to solve. However, as N increases, the total number of possibilities grows too fast and the problem becomes unmanageable without using heuristic search algorithms.

Using permutation encoding, each chromosome says the order of the cities in which the salesman should visit them.

Scheduling of tasks is another niche of application for permutation encoding.

Real-valued encoding

Many optimization problems require tuning a set of real-valued variables. As we previously discussed, binary encoding can be used to represent solutions in a search space of those characteristics. However, it is much more natural to employ real-valued genes. Then, a chromosome in real-valued encoding is nothing but a real-valued vector like $c_a = (1.298, 5.359, 5.127, 1.014)$ or $c_b = (1.137, 5.918, 4.124, 3.069)$.

When we introduce the genetic operators of binary encoding in the next section, it will be clear that the binary crossing operator can be applied to real-valued encoding without any modification and that mutation can be adapted very easily. However, it is also possible to define a different crossing operator only valid for real-valued chromosomes.

This type of representation has lots of straightforward applications. For instance, it can be used to find the dimensions of mechanical pieces, like the blade of a turbine, or to find the optimal separation between radiating elements in an antenna array to match a desired radiation pattern. The possibilities are endless. It can also be used to tune the parameters of another machine learning algorithm, like the weights of a neural network or the bandwidth parameters of mean shift, which is exactly the base of one of the base-station clustering algorithms we will present in the next chapter.

Other encoding formats

After Holland's algorithm was shown to be satisfactory, a great amount of effort was dedicated to generalizing the idea which more complex representations. Nowadays, basically any kind of

data structure can be used if the engineer has enough imagination to create genetic operators which go along the chosen chromosomal representation of the solutions.

In genetic programming, one of the branches of evolutionary computation, tree encoding is generally used. Other problems use many other choices like hashes, objects in object-oriented programming or even indexes to assembly-language instructions have been proposed in the literature with success.

There is no rule-of-thumb to choose an encoding scheme. Whenever we want to design a genetic algorithm, we should think about which encoding seems more natural to our problem and, in case none of the "canonical" representation appears to apply well, we should design our own original scheme. Other issues to be addressed are how to incorporate any constraints in the set of valid solutions onto the representation, whether the order of the genes is relevant or if the number of genes should be constant or variable. The possibilities are endless if we work hard. Indeed, if we are "brave" enough to try it, some schemes even use an adaptive encoding which changes with the generations.

4.6.1.2 Fitness function

The fitness function is also a fundamental piece of any genetic algorithm.

Evolutionary computation methods are completely general, but they need a problem-specific fitness function as an input. In the end, the algorithm will merely optimize the fitness function. Hence, it must reflect the problem's needs as faithfully as possible. Apart from that, the degrees of freedom are enormous: practically any real-valued function defined in the search space can be employed.

For instance, in function optimization problems, the choice for fitness function is obvious: the value of the function to be optimized itself. If we think about the Knapsack problem, the most natural fitness function would be the total value of the items in the bag and for the TSP, the total distance to be traveled seems perfect. When tuning an antenna array to achieve a desired radiation pattern, the mean square error between the desired pattern and the pattern achieved by a certain individual would be an appropriate fitness. It is impossible to cover all the cases which may arise: each specific problems requires a tailored fitness function.

An interesting issue arises when considering constrained optimization problems. There are two main ways to deal which constraints in genetic algorithms.

The first idea is to incorporate the constraints into the fitness function. By including a term which penalizes unfeasible solutions, we can prune the search space and retain only feasible points. However, this has several disadvantages.

On the one hand, the penalization term has to be adjusted properly. If the penalization we include is too small, we risk obtaining invalid solutions as the algorithm's final output. However, if the penalization is too strict, the fitness function becomes non-smooth and we may greatly slow down (or even avoid) convergence in the long term.

Another fundamental problem is the loss of efficiency: along with this idea, we are wasting

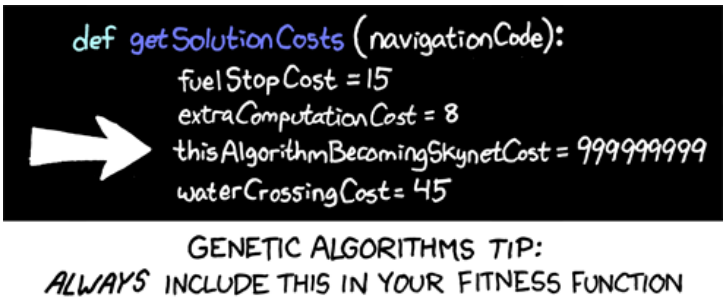


Figure 4.28: Even though it seems to be more natural to program the genetic algorithms to maximize the fitness function, nothing forbids us to try to minimize it. The choice is irrelevant... as long as we are careful enough to be consistent! Otherwise we may create real trouble.

computational resources and genetic variability on invalid solutions. This means that the effective population size can be much smaller than the real population size, so that the performance of the algorithm suffers.

It is much better to avoid exploring unfeasible solutions during the search process at all. This can be done by carefully designing the genetic operators to make the birth of unfeasible individuals impossible or even using a chromosomal representation of the individuals which does not allow unfeasible solutions to be represented. Sadly, this is much easier said than done.

In the end, the most common choice is a trade-off between both ideas. We try to find a representation scheme and a set of genetic operators that make the birth of individuals representing unfeasible solutions as unlikely as possible. Since in many occasions we won't be able to find a way to reduce that probability to 0, we must add also some penalization term in the fitness function to ensure that unfeasible solutions are branded as unfit and are very unlikely to have descendants in the next generation. As long as we keep the probability of obtaining individuals encoding unfeasible solutions low enough, the genetic algorithm will solve the constrained optimization problem satisfactorily.

As a final note, it is important to point out that, in most occasions, fitness evaluation is the computational bottleneck of genetic algorithms. If we think about it, if we have a population size m and simulate T generations, we end up calling the fitness evaluation routine mT times, which may be a pretty big number in many cases. Therefore, optimizing the code of the fitness function as much as possible is really worth the effort.

4.6.1.3 Selection

In order to properly mimic natural evolution, "survival of the fittest" must be incorporated to the algorithm, that is, fitter individuals should have a higher chance of passing its traits onto the next generation.

During this project, we are focusing on generational population models, that is, those in which the individuals live exactly one generation and all of them are replaced by their offspring.

In this kind of genetic algorithms, that is usually implemented by filling a mating pool of size e with individuals picked up from the population. The individuals which end up in the mating pool will be the ones which are allowed to reproduce, giving birth to the individuals which shall populate the next generation of candidate solutions.

Other models allow parents to survive several generations. In those cases, the selection operator acts over the set of individuals formed by the parents and their offspring. In order to mimic generational change, "aging" is included into the fitness evaluation, so that older individuals, that is, those which have already lived through several generations, are penalized in fitness so that they eventually "die" by being removed from the population. Nevertheless, even though that model imitates biology more faithfully, it is seldom used for computational purposes since convergence proceeds more slowly. On the other hand, when genetic algorithms are used by biologists to get insight on natural evolution, which is not our case, this concept becomes attractive.

The pruning of unfit individuals induced by selection is the source of exploitation in genetic algorithms. If the selection step was removed, the algorithm would degenerate into nothing more than a random, brute-force search.

However, we must also be careful to achieve a good balance between exploration and exploitation. If increase the selection pressure too much, that is, we are very strict when evaluating individuals and let only the very best solutions to generate offspring, after a few iterations, the whole population would be filled with very similar individuals. In biological terms, this is denoted as loss of genetic variability, a process that happens for instance in incestuous families. From the point of view of machine learning, this is premature convergence to a local optimum.

Then, we must find a good trade-off in which the search is directed by the most promising individuals but, at the same time, we must allow enough variability so that new, original solutions arise from the evolution process.

Roulette-wheel selection

Roulette-wheel selection is one of the most used selection routines for genetic algorithms. The main idea is very simply: we fill the mating pool with randomly selected individuals from the population, where the probability of picking the i-th individual is proportional to its fitness f_i:

$$p_i = \frac{f_i}{\sum_{j=1}^{m} f_j} \tag{4.84}$$

Note that we are using the convention of fitness maximization. If that was not the case, f_i should be first transformed using a non-decreasing mapping.

We can imagine this selection process as building a roulette wheel divided m portions, one per each individual in the population. The portions are unequal, with an arc length proportional to the fitness of their associated candidate solution. To select an individual, the wheel is spun and when it stops, a marking will tell which was the selected individual. The unequal sizing of the portions is what implements the natural selection principle.

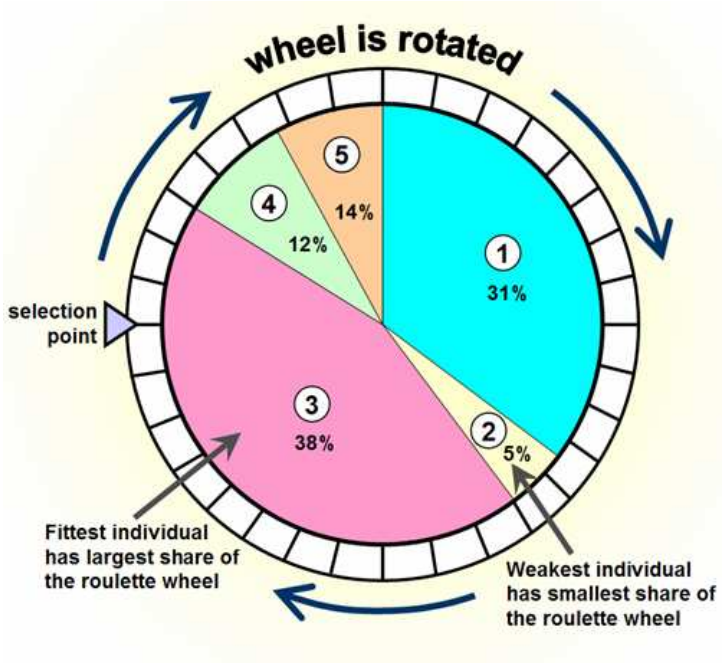


Figure 4.29: A graphical representation of roulette-wheel selection in a dummy example with 5 *individuals.*

The roulette is spun as many times as individuals are to be placed in the mating pool. Even though it may seem strange, it is perfectly possible that some individuals get to place several copies of themselves in the mating pool. That is the computational analogue of having several children with different partners.

Rank selection

Roulette-wheel selection is usually quite effective. However, there are two main scenarios where its performance becomes almost pathological.

On the one hand, both in nature and evolutionary computation, we can witness the birth of super-individuals. Those can be defined as individuals in the population whose fitness in much greater than the average fitness of the population. At first sight, we may think that super-individuals are something good for the species or for the genetic algorithm. However, they also pose a very high risk to the evolution process. Because they are much stronger than the rest of their kind, they have an enormous chance of monopolizing the reproduction process and filling most of the next generation with its own genome. In other words, super-individuals may appear to be good because they are quite fit, but if we leave matters unhandled, they can dramatically reduce the genetic variability.

For instance, let's consider the search space shown in figure 4.27. There, we clearly see two maximums which have a similar height, but one of them is slightly smaller. If we happened to get an individual in the smaller peak, with all the other individuals in points with a much smaller fitness, the next generation would consist of many points around the smaller peak. In

the end, the algorithm would converge to a local maximum instead of converging to the global maximum because a super-individual monopolized the genetic pool.

On the contrary, the complete opposite situation is also troublesome for roulette-wheel selection. If the absolute differences in fitness between individuals of the population are small, all individuals have approximately the same chance of reproducing. This can seem to be a minor problem but, actually, it poses a big threat since it makes the algorithm very sensitive to the absolute values of the fitness function. Even though it is true that a thorough analysis and design of the fitness evaluation routine can solve this issue, it is better to add extra protection. Also, even if we define the fitness function very carefully, this situation will surely arise as we approach convergence. If that's the case, the effect will be that the algorithm progresses more and more slowly as the generations go by and we approach convergence.

If we think about it, both problems are directly caused by equation (4.84). Because the chance of an individual being picked up is directly proportional to the absolute fitness, it is the absolute difference in fitness what determines which individuals are more likely to be picked up for reproduction.

One way to solve this problem is using a rank selection procedure. The idea is analogous to roulette-wheel selection but the probabilities are proportional to the relative fitness instead of the absolute fitness. This means that, for example, it makes no difference whether individual 1 is 100 times fitter than individual 2 or only 0.1% fitter, the selection probabilities would be identical in both cases because, as the name suggests, what determines the selection probability of an individual is its fitness rank within the population. That is, the stronger individual gets the bigger share, no matter what its absolute fitness is and so on.

Mathematically, the m individuals in the population are sorted in ascending fitness order: the weakest individual is assigned index 1 whereas the strongest one is given index m. Then, probability of picking the i-th individual (after sorting) is defined as:

$$p_i = \frac{2i}{m(m+1)} \tag{4.85}$$

Note that $\frac{m(m+1)}{2} = \sum_{i=1}^{m} i$, so that equation (4.85) is simply equation (4.84) where the fitnesses f_i have been replaced with the rank of the individual, that is, its index i within the population in ascending order of fitness.

Scaling methods

An even simpler idea than rank selection, aiming to solve the problems of roulette-wheel selection, is to try to normalize the fitness values.

There are two main ways to do that.

In sigma-scaling, fitness values are normalized by subtracting the average fitness and dividing by the standard deviation of the fitnesses. That is:

$$f'_i = \frac{f_i - \bar{f}}{\sigma_f} \tag{4.86}$$

With $\bar{f} = \frac{1}{m}\sum_{i=1}^{m} f_i$ and $\sigma_f = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(f_i - \bar{f})^2}$.

Another widely used normalization procedure is linear scaling:

$$f'_i = af_i + b \tag{4.87}$$

Where a and b are two user-defined parameters.

In both cases the objective is the same: to moderate the selection pressure so that it is kept relatively constant over the iterations, not too strong at the beginning nor too weak near convergence.

Tournament selection

Tournament Selection is one of the most widely used selection strategies. It works reasonably well in many problems, it is very simple to implement and can be parallelized.

In this selection routine, we first pick randomly two individuals from the population. The probabilities are uniform in this case, that is, $p_i = \frac{1}{m}\ \forall\ i = 1, \ldots, m$. Then, the two chosen individuals are staged to fight a "duel": a random number x is picked from a uniform distribution with support in the interval $(0, 1)$ and it is compared to a prefixed selection threshold, α. If $x \leq \alpha$, the fitter individual wins the duel and gains a place in the mating pool. Otherwise, he surprisingly loses and is the weaker individual the one which will get an opportunity to reproduce. In order for survival of the fittest to apply, the threshold α must be greater than 0.5. The greater it is, the bigger the selection pressure, which gives a direct and simple way for the engineer to tune the balance between exploration and exploitation in the algorithm.

Elitism

As we have argued, in a genetic algorithm, the average fitness usually increases as the generations goes by. However, this is just an heuristic and there is not any guarantee that it will actually be the case. Moreover, if we consider the sequence of fitnesses created by taking the fitness of the fittest individual of each generation, $\{f[t]\}_{t=1}^{T}$. That sequence is supposed to be increasing in the long-term, but it can exhibit some dips. In other words, the fittest individual of generation $t+1$ may be less fit than the fittest individual of generation t. Whether this happens or not is up to the whims of random mutation and crossing.

Because of this, an strategy denoted elitism is usually implemented in genetic algorithms. The concept is quite straightforward: in every iteration, the s fittest individuals are copied onto the next generation without any change. No genetic operator is applied, it is as if they were cloned. The other $m - s$ individuals of the next generation are created using the typical process: selection of individuals to fill the mating pool and then apply the genetic operators.

The usage of elitism guarantees that the sequence $\{f[t]\}_{t=1}^{T}$ is monotonically increasing, a desirable characteristic. Moreover, since the value of s is chosen to be small with respect to m (using $s = 1$ is the most common case) elitism poses no threats to keeping genetic variability and it does not reduce the exploratory power of the algorithm in a noticeable way.

4.6.1.4 Genetic operators

Up to now, we have discussed how to asses the fitness of the individuals in each generation, as well as how to select the ones which will populate with their offspring the next generation of individuals, in a Darwinian survival of the fittest manner. At this point, we must discuss then how do we generate an offspring: this is done by applying the so called *genetic operators*. Those are the source of exploration in genetic algorithms hence they are the last fundamental piece we need to study.

Crossover

The first canonical genetic operator is genetic crossing-over. As the name suggests, this operator is motivated by the biological crossing-over procedure which occurs during the meiosis process in sexual reproduction. Therefore, our objective is to implement routines which, given two individuals characterized by their respective chromosomes (the parents), generate two new chromosomes characterizing two new individuals (the children), by blending the genes in the parent chromosomes in some way. Then, the children will represent candidate solutions which exhibit a mix of the characteristics of the candidate solutions encoded in the parent individuals.

Even though the basic concept is always the same, the way to implement those routines greatly depends on the chosen genetic representation.

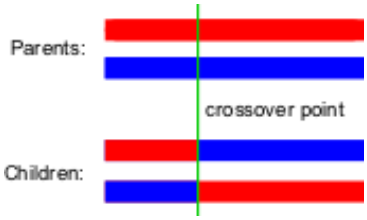
Binary encoding In binary encoding representation, the chromosomes are characterized as fixed-length strings of genes, in this case, binary value genes. A very intuitive way to mimic the meiosis process is to obtain the child chromosomes by interchanging portions of the parents strings. Depending on the way those portions are interchanged we can distinguish several variants.

One-point crossover: A single crossover point p is randomly selected from 1 up to the chromosome length, N. The first p genes of child 1 are the first p genes of parent 1, whereas the first p genes of child 2 will be the first p genes of parent 2. On the contrary, genes $p+1, p+2, \ldots, N$ of child 1 are copied from genes $p+1, p+2, \ldots, N$ of parent 2, and genes $p+1, p+2, \ldots, N$ of child 2 are got from genes $p+1, p+2, \ldots, N$ of parent 1.

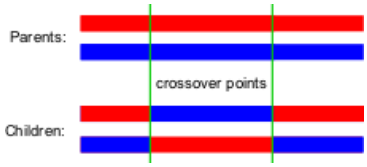
Two-point crossover: We do pretty much the same as in one-point crossover but two random crossover points are selected. The first and last genes of child 1 are obtained from parent 1, and the middle portion of the chromosome from parent 2. As before, child 2 gets the

complementary genes. Note that two-point crossover can be generalized to multi-point crossover simply by picking up n random crossover points.

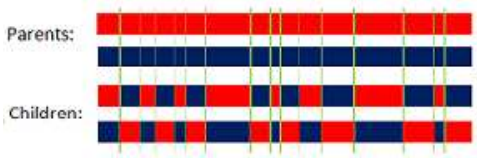
Uniform crossover: A random blending mask is obtained in order to mix the chromosomes. The i-th gene of child 1 is randomly chosen to be either the i-th gene of parent 1 or the i-th gene of parent 2. The other child gets the complementary gene, that is, the one which is not chosen for the first child. If the chromosomes are sufficiently long, each parent contributes to approximately 50% of the genes of each child.



(a) One-point crossover



(b) Two-point crossover



(c) Uniform crossover

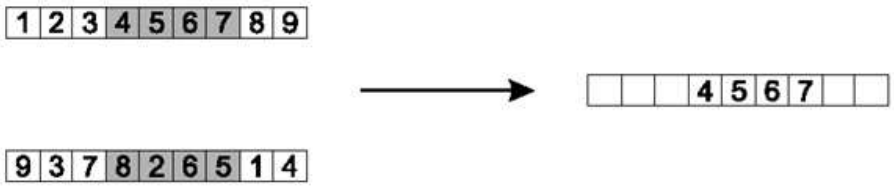
Figure 4.30: Illustration of several ways of applying crossover between binary-encoded chromosomes.

Permutation encoding Ordering problems represented by permutations are actually constrained optimization problems. A chromosome in permutation encoding is a string of natural numbers. However, not any string of natural numbers is valid: there is a constraint imposed on the definition of permutation which established that values along the string cannot be repeated. Because of this, developing crossover operators for permutation encoding is a bit trickier.

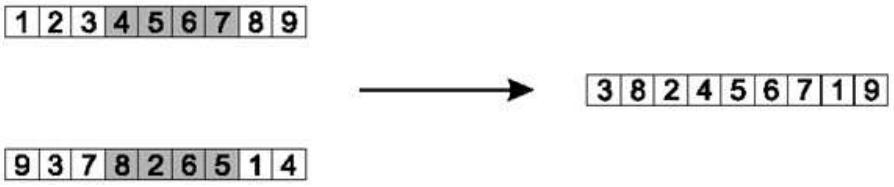
As we already discussed, we could simply apply the straightforward operators of binary encoding and solve the problem of invalid solutions some other way: either by penalizing them with the fitness evaluation step or by adding a second post-processing step which "fixes" invalid chromosomes. However, both ideas are much less efficient that developing customized crossover operators.

Since the family of ordering problems is very broad, lots of effort has been dedicated to this issue and many different crossover operators for permutation encoding exist. Some popular methods are:

Order 1 crossover: This crossover operators tries to preserve the relative order of the elements. In order to generate the first child, we first copy an arbitrary part from the first parent onto the child without change. After that, starting from the cut point of the copied portion of the chromosome, we start copying all numbers which are not in the first portion using the ordering of the second parent and wrapping around at the end. The other child is generated with the same procedure, but parent roles are reversed.



(a) First, we copy an arbitrary part of the first parent onto the first child.



(b) The rest is copied from the second parent, following the order from the cut point, wrapping around and skipping numbers already present.

Figure 4.31: Illustration of order 1 crossover for permutation-encoded chromosomes.

Many other methods exits, like partially mapped crossover (PMX), cycle crossover or edge recombination. The interested reader can learn more about them in the literature, [41].

Real-valued encoding Chromosomes in real-valued encoding are also strings, in this case, with real-valued genes. Because of that, all the crossover operators defined for binary encoding perfectly apply here and, indeed, are employed very commonly.

However, the real-valued nature of the chromosomes opens a new possibility, the so called linear or arithmetic crossover. The idea is also very simple: when the chromosomes belong to a continuous space, such as $\mathbb{R}^N$, a way to "mix" the points is to bring them closer. Specifically, we can generate the children with a pair of convex combinations:

$$\begin{aligned} \mathbf{o}_1 &= \alpha \mathbf{c}_1 + (1-\alpha)\mathbf{c}_2 \\ \mathbf{o}_2 &= \alpha \mathbf{c}_2 + (1-\alpha)\mathbf{c}_1 \end{aligned} \tag{4.88}$$

Where $\mathbf{o}_1$, $\mathbf{o}_2$ are the two children and $\mathbf{c}_1$, $\mathbf{c}_2$ the two parents. The convex combination coefficient α is usually obtained from a uniform distribution with support in the interval $(0, 1)$.

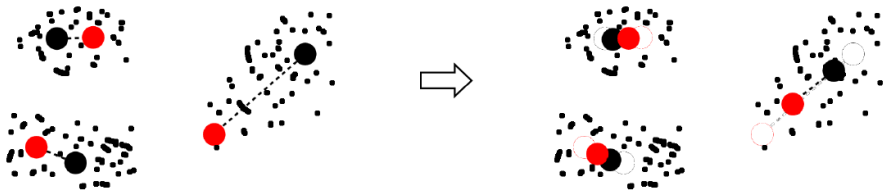


Figure 4.32: Several examples of arithmetic crossover in an scenario with real-valued chromosomes of length 2*.*

Moreover, it is possible to define arithmetic crossover procedures which, instead of mixing the whole chromosome vectors, only act in some portions of the chromosome. In this context, the crossover method represented by equation (4.88) is denoted as *whole arithmetic crossover*. On the other hand, *single-arithmetic crossover* only acts on a randomly chosen gene, being the rest of genes copied from the parents, and *simple-arithmetic crossover* chooses a random crossing point and acts on all genes from the crossing point onwards, being the first genes copied from the parents.

Mutation

Mutation is the other canonical genetic operator used in genetic algorithms. In this case, the inspiration comes from random mutations originated by errors in the DNA replication process which occurs during mitosis or meiosis. In evolutionary computation, we try to mimic those by randomly adding perturbations in the genes of the individuals of the population, just after their "birth". That is, we first obtain the offspring by applying the crossover operator with the individuals in the mating pool and, then, we apply the mutation operator over the offspring to obtain the final individuals which will populate the next generation.

As with crossover, the implementation of mutation differs according to the genetic representation employed.

Binary encoding Since binary-encoded chromosomes only have two alleles, 0 and 1, mutation in this representation is extremely simple: for each gene, we consider a small probability p_m of flipping the bit. Therefore, for a binary chromosome of length N, we just need to throw a biased coin (head probability p_m and tail probability $1-p_m$) per gene and flip the bit of the corresponding gene whenever we get heads. Needless to say, there are very efficient implementations of such a simple routine.

Permutation encoding Mutation in permutation-encoding has the same problems as crossover. If we think about it, the no-repetition restriction imposed in permutation-encoded chromosomes implies that any mutation operation which attempts to produce a valid individual

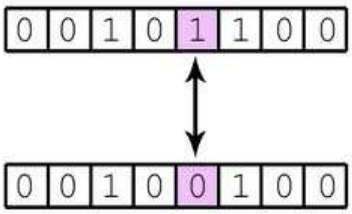


Figure 4.33: An example of mutation of a single gene under binary-encoding genetic representation.

must change at least two genes simultaneously; if we change only one gene, there is no way we can produce a valid chromosome. Because of this, mutation operators for permutation-encoding genetic representation are operators which act in the chromosome as a whole, rather than in a per-gene basis. Similarly, the mutation probability p_m now refers to the probability of the mutation operator being applied to the whole chromosome string, rather than the probability of mutating each gene.

Again, just like with crossover operators for permutation-encoding, there are lots of different mutation operators defined in the literature. Some of them are:

Insert mutation: We randomly pick two genes and move the second to be immediately after the first, shifting the rest of genes accordingly. The biggest advantage of this operator is that it only introduces one change in the ordering, preserving most of both the adjacency and ordering information. In other words, this mutation procedures represents an small jump in the search space for the corresponding ordering problem.

Swap mutation: Again, we randomly pick two genes. However, now what we do is to interchange their positions. The jump induced by this mutation operator in the search space is slightly bigger than the one due to insert mutation. Still, most of the adjacency information is preserved, but ordering information changes.

Inversion mutation: Once more, we randomly choose two genes in the chromosome. This time, all the substring bounded by the two genes has its ordering inverted. This operator preserves adjacency but generates a big jump regarding ordering information.

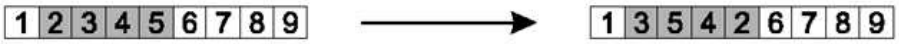
Scramble mutation: This operators randomly chooses a set of genes are, as the name suggests, scrambles the genes generating a totally different ordering. This operator can be used to take big jumps in the search space, allowing to escape attraction from local optimum points.

Real-valued encoding The simplest way to apply mutation in real-valued chromosomes is by using the same procedure as in binary-encoding mutation but, since we do not have binary-valued genes but continuous genes, we modify them by adding a small random number. Then,

(a) Insert mutation example for permutation-encoding

(b) Swap mutation example for permutation-encoding

(c) Inversion mutation example for permutation-encoding



(d) Scramble mutation example for permutation-encoding

Figure 4.34: Illustration of several mutation operators applicable to permutation-encoded chromosomes.

the mutation procedure is to consider, for each gene, a small probability p_m of adding a small random number ϵ drawn from some statistical distribution, usually uniform or Gaussian.

Another possibility is to apply "boundary-mutation". This is useful when the values of the genes are bounded within some interval $\left(g^i_{\min}, g^i_{\max}\right)$. In this case, the i-th gene is replaced, with probability p_m, by either $g^i_{\min}$ or $g^i_{\max}$. Note that this is a very wild mutation, which makes a huge jump in the search space. This can be useful in many occasions, but can lead to a somewhat oscillatory behavior which prevents convergence. Therefore, it is not recommended to use this type of mutation alone. It is best to combine it with smoother mutations, so that we can have both big and small jumps in the search space.

4.6.1.5 Practical implementation issues

In general, genetic algorithms have no closed form: it is up to the engineer how to design them.

For instance, we have discussed several widely used representation formats, together with several genetic operators for each representation. However, many more representations have been described in the literature, coupled with even a bigger amount of genetic operators. As we said before, genetic algorithms are an extremely broad family of algorithms under the basic concept of introducing genetics and natural evolution principles into machine-learning. For each specific scenario, it is possible to create original chromosomal representations and genetic operators which adapt much better to the particular needs of the application. An example of this lies in the next section, where we will discuss several schemes that have been proposed to tackle clustering with evolutionary computing.

Another degree of freedom lies in the amount of genetic operators to be used. We have discussed two of them: crossover and mutation. However, nothing forbids the usage of many more operators simultaneously nor dropping some of them. Indeed, since Holland's SGA, there has been a long debate on the fact of whether mutation and crossover are necessary. In the end,

the current agreement on that issue is as obvious as unhelpful: it depends on each particular problem. However, in general (but not always) it is beneficial to have both since they are complementary.

For instance, let us consider Holland's SGA. In the context of binary-encoding, the crossing operators tend to produce big jumps in the search space whereas the defined bit-flipping mutation operator usually takes smaller jumps, hence it is interesting to combine both. However, if we wanted to use only one operator, we could as long as we think about it carefully. Because the binary-encoding crossover operators cannot change the allele frequencies in the population, it is impossible to explore the whole search space with that family of crossover operators alone. Hence, a version of Holland's SGA with mutation alone could work, but if we use only crossover there is a very big chance of not achieving the global optimum. Nonetheless, for other genetic representations and other implementations of the genetic operators, the situation could be reverted. In the end, we are back to our initial statement: we can design our genetic algorithm almost however we want as long as we think our design throughly.

Moreover, much more exotic changes are available. A particular example is three-parent reproduction which, avoiding dirty jokes, has no other biological inspiration than science-fiction works such as Asimov's "The Gods Themselves". Actually, even n-parent crossover has been considered in the literature. In the end, after we have moved into the domain of machine code, where we have absolute power, we are no longer bounded by nature, are not we?

On a different note, the generic algorithm described in 4.14 contains several conditions and parameters which are somewhat arbitrary.

On the one hand, it is necessary to establish some kind of stopping condition. There are lots of ways to do that: establishing a maximum number of generations to be simulated, fixing a desired fitness level to be achieved by the algorithm or detecting convergence when the average fitness of the population starts to exhibit saturation. However, genetic algorithms are heuristics with no convergence guarantee, so we must take that into account and code some hard limit to avoid entering into infinite loops in a worst-case scenario.

On the other hand, apart from the stopping condition, there are many parameters which need to be chosen: the population size, m; the selection pressure, i.e. the type of selection operator and its parametrization; the crossover probability, p_c; and the mutation probability, p_m. Moreover, those parameters can be fixed or can be adaptively changed as generations go by. As it usually happens in machine learning, little is known yet about the optimal choices and no analytic formulas are available: tuning the algorithm's parameters for each particular application is part of the engineer's duty.

4.6.2 Applications of genetic algorithms in clustering problems

We have introduced genetic algorithms as a very general approach to solve optimization problems. Hence, as long as we can formulate clustering in terms of optimizing a certain objective function, evolutionary computation is applicable to the machine-learning problem we tackle in

this chapter.

However, the adaptation of generic genetic algorithms to the needs of an application like clustering is not straightforward; a lot of work remains to be done.

Recall that a clustering algorithm tries to find a partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ such that $\bigcup_{i=1}^{k} \mathbb{S}_i = \mathbb{X}$ of a certain data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ where $\mathbf{x}_i \in \mathbb{R}^d$, in such a way that all objects in the same cluster are similar and objects belonging to different clusters are dissimilar. In this kind of setup, a candidate solution is nothing but a particular partition $\mathbb{S}[i]$ of the data set. Hence, first of all, a suitable genetic representation of each candidate partition $\mathbb{S}[i]$ needs to be found. Once we have the encoding scheme, the subjective concept of similarity/dissimilarity has to be translated into a quantitative measure which allows to define a fitness function to evaluate each candidate partition, those closer to optimize the objective function induced by the similarity measure will be considered fitter. Furthermore, we also must find a set of genetic operators which allow to evolve the population of individuals throughout generations according to the concepts of evolutionary computation. Finally, as in any genetic algorithm, the selection strategy and the tuning of all relevant parameters needs to be addressed as well.

As usual, there is not a unique, optimum way to solve any of the previous problems. Rather, for each degree of freedom available, literally tenths of different options have been explored in the literature, each with its own set of strengths and drawbacks. During the remainder of this section, our aim is a modest one: to provide a quite brief discussion on some of the most typically used genetic representations and their corresponding genetic operators and to show a couple of simple yet powerful evolutionary clustering algorithms, which allow the reader to get a clearer picture of the situation. This will serve as a solid base to make the genetic base-station clustering algorithms to be presented in the next chapter easily understandable.

Nonetheless, by no means we try to give an extensive review of all the relevant genetic clustering algorithms shown in the literature: this topic is too wide and complex to be treated in this document in further detail. The reader interested on a more exhaustive discussion on genetic clustering algorithms can find in [42] a very nice place to start. In that article, the state-of-the-art in the topic by year 2009 is discussed and references to the articles which propose each of the original algorithms and enhancement to existing algorithms are provided. An even more thorough discussion can be found in [43].

4.6.2.1 Encoding schemes

As we previously discussed, from the times of Holland's SGA, when extremely simple chromosomes consisting of a string of bits were used, lots of methods have been developed to obtain different chromosomal representations in an extremely broad spectrum of radically different applications and scenarios. Nowadays, any data structure imaginable can be used as a chromosome to represent an individual of a genetic algorithm as long as we create genetic operators that go along such genetic representation. Of course, clustering problems are no exception to that statement.

During this chapter, we have already seen several ways to represent mathematically a partition of a data set. Precisely, for a data set with N observations and a partition with k different clusters, we can define a cluster assignation vector $\mathbf{c} \in \{1, \ldots, k\}^N$ such that the i-th sample point belongs to the c_i-th cluster. We also introduced a more redundant representation, the cluster assignation matrix, defined as $\mathbf{E} \in \{0, 1\}^{N \times N}$ such that $(\mathbf{E})^{i,j} = 1$ if the i-th observation belongs to the j-th cluster and $(\mathbf{E})^{i,j} = 0$ otherwise.

Both schemes are widely used yet they are not the only ones.

Binary encoding: The simplest possible form of encoding for a clustering problem is to use a binary string as in Holland's SGA, where the string length N equals the number of individuals in the data set. We can interpret a chromosome of that form in two ways.

If the number of clusters is prefixed to 2, the alleles 0 and 1 can be used to encode each of the two clusters, i.e. the i-th gene of a chromosome will have allele 0 if the candidate partition allocates the i-th observation of the data set to the first cluster, on the contrary, allele 1 would imply that the sample belongs to the second cluster.

For more complex scenarios with more clusters, a medoid-based representation is used: each cluster is characterized by one observation of the data set, generally chosen as the observation belonging to that cluster which minimizes the average dissimilarity to all other objects in the cluster. In other words, clusters are represented by their medoids. In this case, given that certain observations are used to represent their respective clusters, it is usually said that those samples are *prototypes* for their clusters.

Encoding which observations are prototypes and which are not is what represents the partition. Then, the chromosome would have allele 1 in its i-th gene if and only if the i-th observation is a prototype for a cluster in the data set. In order to recover the partition induced for such a representation, observations which are not prototypes themselves are assigned to the cluster represented by the nearest cluster prototype sample. Note that this representation allows a variable number of clusters, which is encoded implicitly as the number of genes in a chromosome with value 1.

Matrix encoding: This form of genetic encoding for clustering problems can actually be regarded as a sort of binary encoding too, because the alleles are binary too. However, the chromosomal structure is arranged as a binary-valued matrix rather than as a vector or string. Of course, since both representations are actually isomorphic, it is not rare that some authors consider this matrix-based encoding within binary encoding methods.

The way to produce chromosomes under this encoding scheme is completely straightforward: take the cluster-assignation matrix $\mathbf{E}$ as we defined during this chapter and we are done.

This method allows a better representation of the partition than medoid-based binary encoding since, for any number of clusters, we can specify the mapping of objects to clusters without having to rely on medoids, which may be inaccurate in some cases since are

limited to prototyping clusters only with points belonging to the data set. In other words, with matrix-based encoding, we can represent all the possible partitions of the data set in k clusters with no exceptions, which is something that can't be done in general with a medoid-based scheme. Moreover, even though it strays from our objective in this project, it is worth noting that matrix-based encoding can be used for overlapping clustering algorithms by allowing each row to have more than one non-zero entry.

The main disadvantages are that the number of clusters, k, has to be prefixed at the beginning and that the memory usage is increased by a factor of k.

Integer encoding: Another straightforward idea: encode each partition using the cluster assignation vector $\mathbf{c} \in \{1, \ldots, k\}^N$ as a chromosome.

This representation has genes with alleles belonging to an alphabet composed of k natural numbers, $\{1, \ldots, k\}$. Recovering the partition is extremely simple: the i-th cluster if formed by all data points j such that $\mathbf{c}^j = i$. The cardinality of the alphabet, k, is what encodes the number of clusters and can be fixed a priori or not depending on the particular algorithm employing the representation. Because the alleles are actually cluster labels, this encoding method is also called label-based representation by some authors. Note also that, for the particular case $k = 2$, integer-based encoding is actually the same thing as the first described case of binary-encoding, interchanging allele 0 by 2.

One of the main characteristics of integer encoding for clustering problems is that the mapping of a particular candidate solution to the genotype induced by the genetic representation is one-to-many. This is actually an obvious property since any permutation of the cluster labels creates a different chromosome encoding the same partition. For instance, $c_a = (1, 1, 2, 2, 3, 3)$, $c_b = (2, 2, 1, 1, 3, 3)$, $c_c = (2, 2, 3, 3, 1, 1)$, $c_d = (1, 1, 3, 3, 2, 2)$, $c_e = (3, 3, 1, 1, 2, 2)$ and $c_f = (3, 3, 2, 2, 1, 1)$ all encode the same partition. This is an undesirable property, since it implies that we are exploring a search space much bigger than necessary. In simple words, this encoding-scheme makes the algorithm waste effort. A way to reduce or even solve this problem is by introducing a renumbering routine which ensures that only one of the $k!$ available representations will be used throughout the search and carefully designing the genetic operators to avoid generating offspring characterized by the discarded representations as much as possible. Further details about this renumbering procedure can be found in [43].

A radically different but, in my opinion, less interesting way of encoding partitions with an array of integer numbers is by using again a medoid-based prototype representation. The idea is totally analogous to the one presented in the binary-encoding section but, instead of using an array of N binary numbers with k ones, encoding the position of the prototypes medoids within the data set, we encode the positions using an integer-valued string of length k, which contains the indexes of the data-points which would correspond to a binary allele 1 if binary-encoding was used. As a simple example to illustrate the

difference, consider a data set with five observations such that points 2 and 4 are the medoid prototypes, a binary-encoding chromosome would be $c = (0, 1, 0, 1, 0)$ whereas the corresponding integer-encoding chromosome would be $c = (2, 4)$.

Needless to say, all the disadvantages of medoid-based representation; such as the inherent inaccuracy and lack of expressive power due to being forced to describe clusters only with points in the data set, assigning the rest of samples based only on the similarity with the reduced set of prototypes, are also present in this medoid-based integer encoding.

Real encoding: Real-valued encoding has already been discussed for more general problems: we use chromosomes which are real-valued vectors.

Real-valued encoding is also used with prototype-based representations of the partitions. However, because with real-valued genes we can actually encode features in the sample space, we are no longer limited to using prototypes which have to belong to the data set. For instance, real-valued encoding allows to represent clusters by the geometric median or, most commonly, by the cluster centroids.

The most common format is to append each of the cluster prototypes, be it centroids, geometric median or whatever any other think we like, in a single real-valued array. Since the prototypes belong to the same space as the observations, each of them is a d dimensional vector. Hence, a chromosome containing all prototypes one after the other corresponds to a Nk-dimensional vector. Mathematically, if we have a set of k prototypes, one per cluster, $\{\mathbf{y}_j\}_{j=1}^{k} \mid \mathbf{y}_j \in \mathbb{R}^d$, then the corresponding chromosome becomes $\left[\mathbf{y}_1^T \mathbf{y}_2^T \dots \mathbf{y}_k^T\right]$.

4.6.2.2 Genetic operators

Obtaining genetic operators for clustering problems is an extremely pedagogical example for people willing to get deeper into genetic algorithms. Courses on evolutionary computing which serve as a gentle introduction to genetic algorithms usually provide examples for very simple cases like the Knapsack problem, TSP or optimization of a scalar, real-valued function. In those cases, both the encoding schemes and the genetic operators are very easy to design, since there is a very obvious, optimal choice and few complications arise. Nonetheless, clustering problems are an advanced topic which serves to illustrate how genetic algorithms can be really powerful yet highly complicated to design.

The key concept here is *task-orientation*. All the genetic operators discussed in section 4.6.1.4, except maybe those related to permutation encoding, modify and mix the chromosomes in the population without any consideration on the meaning which was encoded by the genes being altered.

Imagine that we are trying to optimize the design of the blade of a rotor. Suppose that each of the genes controls a certain parameter or dimension of the design. It is plausible that the variables are interrelated, that is, if the dimension of some part of the piece is a, that may

impose a constraint on the feasible values for another dimensions. In other words, not every point in $\mathbb{R}^N$, assuming N to be the number of variables to be optimized, is feasible.

Furthermore, consider that we use the uniform crossover operator for real-valued chromosomes as defined in section 4.6.1.4. Recall that such operator generates the offspring by randomly mixing the alleles of both parents. Then, it is perfectly possible that such an operator produces an unfeasible children out of two feasible parents.

We already argued that, by properly designing the fitness evaluation function to filter out unfeasible solutions, a genetic algorithm using unspecialized genetic operators will work. Nonetheless, it is easy to see that we can do much better than that by defining better operators. We will say that a genetic operator is *task-oriented* when the genes are mixed or modified in a way that bears a certain meaning in the specific context for which the genetic algorithm is intended. In particular, for a clustering problem, a genetic operator is *cluster-oriented* when it splits, merges, recombines, duplicates or deletes the clusters represented by the chromosomes being mixed or modified.

In scenarios with encoding schemes which are one-to-many, such as label-based encoding for clustering, we may have two different chromosomes which actually represent the same solution to the problem. If two parents encoding the same solution were to have offspring, the most reasonable expectation is that the children would also encode that solution, because the mixture of two parents which the same meaning should keep such a meaning. This leads to define the concept of *context-sensitivity* for crossover operators, introduced in [43].

A crossover operator is said to be *context-sensitive* if it it task-oriented and the offspring generated by two (possible different) chromosomes encoding the same candidate solution always produce offspring which encode the same solution too.

It is very easy to see that, for all the genetic representation schemes for clustering problems discussed in the previous section, traditional crossover operators, such as n-point crossover, uniform crossover over arithmetic crossover are context-insensitive. As an example, let as consider one-point crossover under label-based encoding. Imagine that we have a dummy example with only six points, and two solutions which are to be recombined. Let the first parent chromosome be $c_a = (1, 1, 2, 2, 3, 3)$, and the second $c_b = (1, 1, 3, 3, 2, 2)$. Note that both parents actually encode the same partition, the only difference is that labels were swapped. Assume that, the crossover point is randomly chosen to be the fifth gene. Then, the generated offspring would be $o_a = (1, 1, 2, 2, 2, 2)$ and $o_b = (1, 1, 3, 3, 3, 3)$,which clearly encode a partition different than that of their parents. Besides, in the particular case that the number of clusters was prefixed to 3, the creation of an empty cluster in each child would have given raise to two unfeasible solutions. An exhaustive set of examples to prove the context-insensitivity of traditional crossover operators under the other genetic representation schemes for clustering problems previously discussed can be found in [42].

To summarize, generic genetic operators blindly manipulate genes without considering the relationship existing between genes nor the implications of the modifications being carried out.

This is terrible for clustering because, for all the genetic representation schemes discussed, it is actually the set of interconnections between genes what encodes the partition and, thus, the real optimization goal lies in those interrelations. As a consequence, the development of cluster-oriented genetic operators is a major concern in the design of genetic clustering algorithms.

As a matter of fact, genetic operators usually constitute the major point of innovation in most articles claiming to propose an original genetic clustering algorithm. Because of this, there are simply too many options, all of them worthy of attention, to be discussed in this document. Our aim is not an exhaustive study of the applications of genetic algorithms for clustering but, rather, providing a gentle introduction to use them later as a tool for our particular application. Therefore, we have decided to avoid an enumeration of cluster-oriented genetic operators. Instead, in the next sections, we will show two examples of genetic clustering algorithms and their corresponding genetic operators will serve as an illustration to complement this discussion. The reader who is willing to know more about cluster-oriented operators is referred to [42] and [43].

4.6.2.3 Genetic K-Means applied to spectral clustering

Within the very broad spectrum of clustering algorithms which use evolutionary computation, we can roughly identify two main types.

On the one hand, we have those which use a genetic algorithm to enhance another existing clustering algorithm, to tune their parameters or to optimize the randomized aspects of the algorithms. In many cases, this family of algorithms end up using traditional genetic operators since the genetic algorithm is not the one which actually obtains the partitions but, rather, influence the result in an indirect way, that is, by acting in some way on the main clustering algorithm being used.

The example we discuss in this section is of this kind. We will see how a genetic algorithm can be used to solve one of the main weaknesses of K-Means: its extreme dependence on the random initialization of the centroids. In [44], the authors propose what they call a "genetic spectral clustering algorithm". Nevertheless, what they actually do is to design a K-Means algorithm which uses evolutionary computation to improve its robustness against the random initialization step. After that, they use a completely traditional spectral clustering algorithm, such as any of the ones discussed in section 4.4.5, where the K-Means rounding step employs that genetically-enhanced version of K-Means.

The idea of using genetic algorithms to improve standard K-Means is by no means an original contribution of [44]. Many articles have been published pursuing the same objective. Probably the first article introducing the concept was [45], with their Genetic K-means Algorithm (GKA), which was published in 1999. Their work was vastly improved by [46] in 2004, with the introduction of an algorithm which exhibits essentially the same performance but which executed much faster, hence, their algorithm was called Fast Genetic K-means Algorithm (FKGA). Since then, tenths of versions which introduce some modifications have appeared up to the point that even quantum versions exist as in [47].

Nonetheless, as most of the aforementioned algorithms are quite similar, we decided to follow [44] mainly because of its simplicity. The main choices of the algorithm from the point of view of evolutionary computing are:

Genetic representation: Given that K-means inherently represents clusters by their centroids, the most appropriate encoding scheme is real-valued encoding with centroid-based prototyping. Therefore, we will be working with chromosomes which are Nk-dimensional real-valued vectors $\mathbf{c}$.

Fitness function: In [44], the Rand Index is used as fitness function. However, evaluation of the Rand Index in this context requires a set of labels available; otherwise it is nothing but a measure of consistency between two partitions. Given that clustering is, by definition, an unsupervised problem, assuming that such a labeled set is available does not seem very reasonable in my opinion. As a consequence, we substitute the fitness function by the one used in [45]:

$$F(\mathbf{c}) = \begin{cases} \bar{C} + b\sigma_C - C(\mathbf{c}) & \text{if } C(\mathbf{c}) \leq \bar{C} + b\sigma_C \\ 0 & \text{otherwise} \end{cases} \tag{4.89}$$

Where $C(\mathbf{c})$ is the K-means cost function as defined in (4.11) evaluated at the partition $\mathbb{S}$ induced by the set of centroids encoded in the Nk-dimensional real-valued chromosome $\mathbf{c}$; $\bar{C}$ is the average K-means cost of the population, $\bar{C} = \frac{1}{m}\sum_{i=1}^{m} C(\mathbf{c}[i])$; σ_C is the standard deviation of the K-means cost of the population, $\sigma_C = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(C(\mathbf{c}[i]) - \bar{C}\right)^2}$; and b is an scaling tunable parameter which ranges usually between 1 and 3 according to the authors.

Selection: Basic roulette-wheel selection with a generational population model (offspring completely replaces parents) is employed.

Genetic operators: Whole-arithmetic crossover is used for chromosome recombination to produce offspring. The mutation operator is applied independently with probability p_m to each gene. In case of a mutation actually occurs, the value of the corresponding gene is totally erased and substituted by a random value obtained from a uniform distribution between the minimum and maximum value that can be attained for that particular gene. A good way to fix that range is by using the range of the corresponding feature in the data set.

The algorithm 4.15 summarizes a basic version of the GKA.

When we discussed standard K-means we said that, given the high dependence it exhibited on the random initialization step, it was a common practice to execute several instances of the algorithm in parallel and keep the one which achieved better results. Actually, algorithm 4.15 is doing nothing but an improved version of that idea: instead of a brute-force random search

Algorithm 4.15: Basic Genetic K-means Algorithm

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \mid \mathbf{x}_i \in \mathbb{R}^d$
input : Desired number of clusters k
input : Population size m and mating pool size e
input : Crossover probability p_c and mutation probability p_m
output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of $\mathbb{X}$

Randomly initialize m different sets of centroids $\left\{\boldsymbol{\mu}_j[i]\right\}_{j=1}^{k} \ \forall\, i = 1, \ldots, m$ to create a population of m individuals $\mathbf{c}[i] = \left[\boldsymbol{\mu}_1[i]^T \ldots \boldsymbol{\mu}_k[i]^T\right]$;
while *convergence criterion not satisfied* **do**
  for $j \leftarrow 1$ **to** m **do**
    Run standard K-means algorithm using the set of centroids encoded in $\mathbf{c}[i]$ as initial centroids ;
    Replace $\mathbf{c}[i]$ by the set of centroids obtained after convergence of K-means ;
    Evaluate the fitness of the new $\mathbf{c}[i]$;
  end
  Apply selection to fill a mating pool of size e with individuals from the population ;
  Shuffle the mating pool and divide it in $e/2$ pairs of individuals ;
  Apply crossover to each pair with probability p_c to obtain two new individuals. Otherwise, copy the original pair as if they were the children ;
  Apply mutation to each individual gene of each individual in the offspring with probability p_m ;
  Replace the current generation with the offspring ;
end
Cluster the data set $\mathbb{X}$ using as initial centroids those encoded in the fittest individual of the last generation to obtain the output partition $\mathbb{S}$;

on the space of initial centroids, we employ genetic algorithms to make the search much more efficient.

Several aspects in algorithm 4.15 could be different according to the particular version of genetic K-means we use.

For instance, the original GKA by [45] used a label-based encoding instead of the real-valued, centroid-based representation we have shown. They discard the use of a crossover operator, however, they also define a cluster-oriented mutation operator which modifies the genes with a probability proportional to the distance between the data-point corresponding to the gene and its currently associated centroid. In that way, points which are poorly assigned because the square error is high have a great probability of mutation. Also, if a mutation occurs, the new allele is not chosen from a uniform distribution but, rather, from a distribution weighted proportionally to the distances between the sample and the rest of centroids.

Nevertheless, in the end, the basic idea is similar throughout the different versions.

Unlike standard K-means, it can be proven that the introduction of this evolutionary search guarantees convergence to the global minimum of the K-means cost function in the long term. Nonetheless, the price to pay is obvious: the computational complexity greatly increases.

4.6.2.4 Fast Evolutionary Algorithm for Clustering (FEAC)

On the other hand, the other main family of genetic clustering algorithms are those which use genetic algorithms as the core of the clustering algorithm itself.

This kind of algorithms are much more dependent on the genetic representation and the usage of cluster-oriented operators becomes a must. On the other hand, since they are not subject to the limitations imposed by employing a simpler algorithm, like K-means, as the main tool to generate partitions, this type of genetic clustering algorithms have the potential to outperform those which only use evolutionary computation to enhance simpler clustering algorithms, at the price of a considerable increase both in complexity and execution time.

To illustrate an algorithm of this kind, we will discuss a genetic clustering algorithm presented in [48]. The authors named it Fast Evolutionary Algorithm for Clustering (FEAC). Its main characteristics are:

Genetic representation: Label-based encoding is employed with no restrictions imposed on the number of clusters k. Different individuals in the population may represent solutions with different numbers of clusters.

Population initialization: For each individual, we randomly select an initial number of clusters k_i between 2 and $\sqrt{N}$, being N the number of observations in the data set. After that, each data point is assigned to a cluster at random, that is, each gene in the chromosome is randomly assigned a value in the set $\{1, 2, \ldots, k_i\}$.

This initialization populates the first generation of individuals with partitions representing different number of clusters. This favors diversity, speeding up the convergence time. However, the usage of a unique, fixed number of clusters k during the initialization phase would also work since the adaptive nature of the algorithm would end up finding the "correct" number of clusters anyway.

Fitness function: The silhouette function as defined in section 4.1.2 is employed. However, the authors introduce two interesting ideas.

On the one hand, they comment on the possibility of expressing the overall fitness of an individual as a weighted sum of per-cluster fitness values. Because of the way the silhouette function is defined, doing that is straightforward. If we let $s(i)$ be the silhouette coefficient of the i-th observation $\mathbf{x}_i$, then the silhouette-fitness of cluster $\mathbb{S}_j$ can be expressed as:

$$F_c(\mathbb{S}_j) = \frac{1}{|\mathbb{S}_j|} \sum_{\{i|\mathbf{x}_i \in \mathbb{S}_j\}} s(i) \tag{4.90}$$

The total fitness of the individual can be then computed as:

$$F(\mathbb{S}) = \frac{1}{N} \sum_{j=1}^{k} |\mathbb{S}_j| F_c(\mathbb{S}_j) \tag{4.91}$$

Which is completely equivalent to evaluation of the silhouette function for the partition $\mathbb{S}$ encoded by the individual.

Doing this is interesting because, as we are about to see, the authors introduce those per-cluster fitness evaluations to allow their cluster-oriented mutation operators to modify less fit clusters with a greater probability than those clusters which are fitter.

Another interesting concept is the usage of a *simplified silhouette function* in order to reduce computational complexity. The idea is straightforward though: we substitute, in the definitions of $a(i)$ and $d(i, \mathbb{S}_l)$ needed for evaluation of the silhouette coefficients $s(i)$, the dissimilarity averages by simply the dissimilarity between the observation and the cluster centroid. Then, $a(i)$ becomes the dissimilarity between the i-th observation and the centroid of the cluster to which it belongs and, similarly, $d(i, \mathbb{S}_l)$ is now the similarity between the i-th observation and the centroid of cluster $\mathbb{S}_l$.

According to the authors, the introduction of this simplified silhouette function barely produces a degeneration of the algorithm's overall performance while it speeds up the fitness evaluation step considerably in large data sets.

The authors in [48] also propose a novel per-cluster Rand Index. However, as we previously argued, given that we consider that no-access to a labeled set is available, we won't think it to be useful for our application.

Selection: Roulette wheel selection or rank-selection are proposed. Moreover, elitism is employed so that the fittest individual is allowed to continue living in the next generation without any modification.

Genetic operators: The most interesting point is undoubtedly the novel cluster-oriented genetic operators that the authors define.

The first shocking point is their decision to completely discard the usage of any crossover operator; their genetic algorithm is mutation only. As we discussed before, that is a perfectly valid choice as long as the mutation operators are chosen in a way that they can explore the search space both with big and small jumps.

Precisely, they define two mutation operators, denoted by the extremely original names of Mutation Operator 1, (MO_1), and Mutation Operator 2, (MO_2).

Mutation Operator 1 is essentially a merging operator. It can be applied only in individuals which represent a partition with at least 3 clusters, since the authors consider that at least two non-empty clusters should be produced by the algorithm. The pseudo-code for the operator is quite simple:

Algorithm 4.16: FEAC Mutation Operator 1

input : An individual $\mathbf{c}[i]$ encoding a candidate partition $\mathbb{S}[i]$

if $|\mathbb{S}[i]| > 2$ **then**
  Randomly select a number of clusters to be mutated n in the set $\{1, 2, \ldots, |\mathbb{S}[i]| - 2\}$;
  for $j \leftarrow 1$ **to** n **do**
    Randomly choose a cluster $\mathbb{S}_u[i]$ of the partition $\mathbb{S}[i]$;
    Assign each observation $\mathbf{x}_a$ belonging to cluster $\mathbb{S}_u[i]$ to the nearest remaining cluster $\mathbb{S}_l[i] \neq \mathbb{S}_u[i]$ in partition $\mathbb{S}[i]$ according to the distance between observation $\mathbf{x}_a$ and each of the cluster centroids $\boldsymbol{\mu}_l$.
  end
  Update the individual $\mathbf{c}[i]$ so that it now encodes the newly obtained partition ;
else
  Individual $\mathbf{c}[i]$ is not mutated ;
end

As we can see, the mutation operator 4.16 merges a random number of clusters in the partition all together to generate the new individual, ensuring that at least 2 different clusters remain. The merging is carried out by eliminating complete clusters from the partition and placing the objects which belonged to those clusters within the remaining clusters. Each object is reassigned to the cluster whose centroid is closer. However, this could be easily generalized by a minimize average-dissimilarity criterion in case that, due to the topology of the data set, centroids were not good cluster prototypes.

On the other hand, Mutation Operator 2 is created to be a splitting operator. Its pseudo-code is:

Mutation Operator 2 represents the opposite task than Mutation Operator 1. It mutates an individual by splitting a random number of clusters. The way to split the clusters chosen by the authors is based on geometric principles: an observation $\mathbf{x}_a$ belonging to the cluster is randomly chosen as seed for one of the new clusters, and the observation in the same cluster furthest apart from $\mathbf{x}_a$ is chosen as the seed for the other cluster to be created. The rest of objects in the cluster to be split are allocated to one or another depending on which of the two seeds is closer. Because we can only split clusters with at least 2 objects, singletons are ignored by mutation operator 4.17.

Again, it is possible to straightforwardly generalize this idea proposed in [48] to work in cases where Euclidean metric assignments is not appropriate by changing the Euclidean

Algorithm 4.17: FEAC Mutation Operator 2

input : An individual $\mathbf{c}[i]$ encoding a candidate partition $\mathbb{S}[i]$

Randomly select a number of clusters to be mutated n in the set $\{1, 2, \ldots, |\mathbb{S}[i]|\}$;
for $j \leftarrow 1$ **to** n **do**
    Randomly choose a cluster $\mathbb{S}_u[i]$ of the partition $\mathbb{S}[i]$;
    if $|\mathbb{S}_u[i]| > 2$ **then**
        Randomly choose an observation $\mathbf{x}_a$ belonging to cluster $\mathbb{S}_u[i]$;
        Pick up the observation $\mathbf{x}_b$ belonging to cluster $\mathbb{S}_u[i]$ which is furthest apart from $\mathbf{x}_a$;
        Split cluster $\mathbb{S}_u[i]$ in two creating a cluster $\mathbb{S}_{u,1}[i]$ which contains all observations in $\mathbb{S}_u[i]$ which are closer to $\mathbf{x}_a$ than to $\mathbf{x}_b$ and another cluster $\mathbb{S}_{u,2}[i]$ with the observations closer to $\mathbf{x}_b$ than to $\mathbf{x}_a$;
    else
        Do not split cluster $\mathbb{S}_u[i]$ encoded in $\mathbf{c}[i]$;
    end
end
Update the individual $\mathbf{c}[i]$ so that it now encodes the newly obtained partition ;

distance metric assignation to the minimization of an arbitrary dissimilarity function.

Combined, both mutation operators allow to modify the shapes of the clusters, relocating observations from one cluster to another thus effectively changing the partitions. Moreover, they are even to modify the total number of clusters k encoded in each candidate partition in an adaptive way. Because the number of clusters to be modified each time an individual is mutated is randomly chosen between only 1 to almost the entire partition, some individuals will experience mutations which will strongly modify their genome, providing a large jump in the search space, whereas others will be slightly affected, representing a small step. This generates a diverse population less susceptible to get stuck at local optimum points.

Therefore, together, both mutation operators provide enough exploratory power for the genetic clustering algorithm to be able to find the optimal partition of the data set as generations go by.

FEAC as described in [48] does not employ any mutation probability: all individuals suffer either a mutation due to Mutation Operator 1 or Mutation Operator 2. Which operator is to be applied to each individual is randomly decided with probabilities p_1 and $p_2 = 1 - p_1$. In the most basic version, the authors use $p_1 = p_2 = 0.5$, that is, both mutation operators are equiprobable. However, they also discuss an improvement consisting of introducing an adaptive behavior: the algorithm keeps track of how well each mutation operator performs in our data set by measuring the average increase (or decrease) in fitness of all individuals

that have been mutated according to each of the two operators. In the next iteration, the probabilities p_1 and p_2 are computed in a way that the mutation operator which performed better is assigned a bigger probability of being applied.

Another interesting point which they discuss is how to select, within each mutation operator, the clusters to be modified. In the most basic version, clusters are picked up with equal probabilities. Nonetheless, they realized that, if they were able to define per-cluster fitness values as we previously discussed, it would make sense that less fit clusters have a greater probability of being chosen for modification. Assuming, as it is the case for the silhouette fitness function, that the per-cluster fitness values $F_c(\mathbb{S}_j)$ exist within the closed interval $[0, 1)]$, they defined a roulette-wheel cluster selection procedure to pick which n clusters to be modified during the execution of the mutation operators, with associated probabilities:

$$p_j = \frac{1 - F_c(\mathbb{S}_j)}{\sum_{i=1}^{k} 1 - F_c(\mathbb{S}_i)} \tag{4.92}$$

In the end, even though a huge number of genetic operators have been proposed in the literature for clustering problems, we believe that FEAC's mutation operators provide an excellent introductory example on how to create cluster-oriented genetic operators to obtain a high-performance genetic clustering algorithm.

FEAC is outlined in algorithm 4.18.

As we can see, FEAC still employs K-means in its formulation. Nonetheless, the fundamental difference with respect to GKA lies in the role of the genetic part of the algorithm within the entire routine.

GKA was essentially K-means and the genetic algorithm merely tried to obtain different replicated through natural evolution rather than random search.

On the other hand, in FEAC, the genetic algorithm is the one which actually generates each of the candidate partitions in the population. K-means is used as yet another operator, in this case, it is employed as a local search procedure which fine-tunes the partitions obtained by the genetic-algorithm prior to fitness evaluation.

The authors argue that they chose K-means as the fine-tuning tool in their algorithm because they felt that it provided an interesting synergy with their cluster-oriented genetic operators.

Since operators 4.16 and 4.17 are able to modify the number of clusters through an evolutionary search, the limitation of using a fixed number of clusters k in K-means is overcome. Moreover, since those genetic operators also create rough partitions of the data set, the set of initial centroids for the K-means local search step is constantly evolved.

On the other hand, K-means allows to fine-tune the partitions obtained by the genetic algorithm by ensuring that clusters are compact and that every observation is assigned to the cluster whose centroid is closest; in other words, the K-means step ensures that the silhouette

Algorithm 4.18: Fast Evolutionary Algorithm for Clustering (FEAC)

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \mid \mathbf{x}_i \in \mathbb{R}^d$
input : Population size m
output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of $\mathbb{X}$

```
Initialize a population of m individuals at random ;
while convergence criterion not satisfied do
    for j ← 1 to m do
        Compute the centroids for the partition encoded by the j-th individual ;
        Run standard K-means algorithm using those as initial centroids ;
        Update the individual with the partition that has been obtained with K-means ;
        Evaluate the fitness of the new individual ;
    end
    Apply selection to fill a mating pool of size m with individuals from the population ;
    Apply the cluster-oriented mutation operators to the individuals in the mating pool
    to obtain the offspring ;
    Replace the current generation with the offspring. Leave the fittest individual
    unchanged (elitism). ;
end
The output partition is obtained as the partition encoded by the fittest individual of the
last generation ;
```

fitness function is always non-negative throughout the algorithm. Also, if the empty-cluster handling criterion of K-means is set to drop the cluster, the authors point out that K-means can also speed up the convergence to partitions with the optimal number of clusters.

In the end, using K-means as yet another lesser operator and letting the evolutionary search by the pillar of the clustering algorithm allows to outperform many alternative clustering algorithms, even those which use evolutionary search to fine tune another clustering algorithm, such as GKA. However, we must realize that the computational cost involved is brutal, even for the standards of other genetic clustering algorithms like GKA. Therefore, despite the excellent quality of the achieved partitions, FEAC can only be used on a handful of selected problems which have an offline processing nature and very loose time constraints.

Chapter 5

Clustering algorithms for BTS coordination in cellular environments

5.1 Introduction

The fundamental ambition of this project is motivating new research lines to improve current BTS coordination schemes for MIMO cellular systems by creating clustering algorithms which dynamically group BTSs to optimize the performance of the system. The path towards that aim has been arduous, given the vast amount of prerequisites needed to tackle the problem.

During chapter 2, the concept of MIMO communication systems was introduced. There, we discussed the generic MIMO system model and particularized it to study the main canonical MIMO scenarios that arise when considering the special needs of each application. We also saw that an OFDM-based cellular system such as any 4G or 4.5G radio access network could be modeled as a distributed multi-user MIMO scenario.

A standard, single-user MIMO system was considered to be a communication system in which the transmitter and the receiver had t and r antennas respectively. We assumed that, due to the usage of OFDM, the signal experiences slow, flat fading and thus the propagation between any transmit-receive antenna pair could be modeled by a random attenuation coefficient and a phase-shift, that is, as a multiplication by a complex scalar coefficient h_{ij}. By collecting all the h_{ij} in a $r \times t$ matrix $\mathbf{H}$, we could model the effect of the channel on the whole transmission. Under the common restriction of having more transmit than receive antennas, $t \geq r$, such a system can handle the transmission of up to r data streams in parallel, disregarding the small probability of dealing with a rank-deficient channel matrix $\mathbf{H}$. When $\mathbf{H}$ is not a diagonal matrix, this implies that each data stream experiences a high amount of interference from the other data streams. As a consequence, it was necessary to include signal processing in the transmitter and the receiver to try to reduce that phenomenon. We considered linear spatial precoding in the transmitter by means of a $t \times r$ matrix $\mathbf{W_{tx}}$ and linear spatial filtering in the receiver with a $r \times r$ matrix $\mathbf{W_{rx}}$. This led to the generic system model shown in equation 2.2.

The most surprising conclusion of chapter 2 was that a cellular system could be modeled pretty much the same way. Focusing on a particular OFDM slot, we have a set of M cells; equipped with one BTS and serving one user each. Moreover, we consider that BTSs have t transmit antennas and UEs have r receive antennas. There are two main approaches to model this scenario: as a set of M, t-by-r interfering point-to-point MIMO systems or as a distributed multi-user MIMO system with Mt transmit antennas and Mr receive antennas which are scrambled all over a wide geographical area.

The first model has the obvious advantage of being much simpler to implement. However, as we discussed during this project, the interference existing between the M parallel point-to-point MIMO systems degrades performance down to levels which are unacceptable if we want to satisfy the demands of the next generation mobile communication systems such as 4.5G.

As a consequence, our research is focused on the second model. However, we must also be realistic: treating a cellular system as a gigantic distributed multi-user MIMO system is something that can not be implemented without incurring in an unfeasible cost. Estimating the complete $Mr \times Mt$ channel matrix would require an unrealistic amount of pilots. Besides, implementing a signal processing scheme which involves the coordination of M base-stations or even M UEs is completely out of question.

As we discussed in chapters 1 and 2, in order to deal with that, we consider something between both approaches: by grouping cells in subsets or clusters, we have a set of S distributed multi-user MIMO systems, each having a relatively small number of cells $\{L_i\}_{i=1}^{S}$. This keeps inter-cell channel estimation and BTS coordination within a manageable complexity for each of the S clusters while, at the same time, the interference levels between cells are much lower than without any coordination at all.

Once the packing of base-stations into clusters has been chosen, we can estimate the corresponding within-cluster channel matrix $\mathbf{H}_i \in \mathbb{C}^{L_i r \times L_i t}$ and calculate the precoder matrix $\mathbf{W_{tx}}$ and filter matrix $\mathbf{W_{rx}}$ using the techniques developed during chapter 3. However, note that even if the size of the channel matrix we have to deal with has been considerably reduced, we are still dealing with a distributed multi-user MIMO scenario. Moreover, UE coordination is generally regarded as unfeasible. Therefore, we need to take into account both facts and select, within the set of techniques available, those which consider PBPC and don't need coordination between receivers in the downlink or between transmitters in the uplink.

The usefulness of this scheme is obvious and, because of that, a great deal of research has been developed around this topic. However, as we argued in chapter 1, even though a lot of effort has been dedicated to the design of the signal processing algorithms, the fundamental issue of how clusters are made seems to be completely ignored. Since universal frequency reuse has been proposed for 3G mobile communication systems, grouping base-stations together to form clusters has no longer been a problem of interest... hopefully this project will set the basis to change that.

Probably, to find the nearest precedent when BTSs were allocated to clusters, we need to go

back to the old times of GSM. However, clusters were made under radically different needs.

GSM employed a combined TDMA/FDMA scheme for multiple access: a set of 200 KHz channels are created within the available GSM bands (FDMA) and each channel is subsequently divided in 8 different time slots (TDMA). However, GSM does not employ universal frequency reuse. Rather, channels are assigned to base-stations trying to leave guard bands between channels belonging to neighboring stations. Nevertheless, because the amount of channels is finite and every BTS usually requires several channels to carry the traffic it will be offered during the busy-hour, it was necessary to introduce some kind of frequency reuse: and that is how the concept of base-station clusters initially appeared.

GSM models BTS deployments over a planar surface as a regular hexagonal grid.

First of all, even though the coverage of a cell is approximately circular considering isotropic transmit antennas, it is not possible to cover completely surface using circles without overlapping. To simplify the model, polygonal coverage zones were employed instead. Moreover, hexagonal cells were chosen since hexagons are the polygon which maximizes the area-to-radius ratio, hence less cells are needed to cover a given surface.

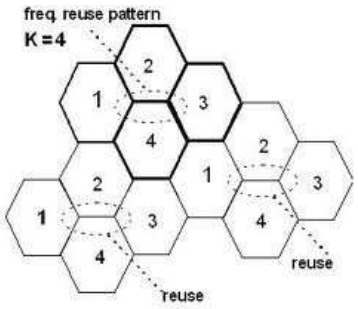
In order to create what was denoted as cluster at the time, several hexagonal cells were grouped in regular structures, in such a way that the whole coverage area could be spanned by a systematic translation of the cluster structure. Some examples with several different cluster sizes are shown in figure 5.1.

The complete set of available GSM channels is then distributed between the BTSs in each cluster in such a way that every channel is assigned uniquely to one and only one BTS within the cluster. Thanks to that, interference between cells in the same cluster was reduced to negligible levels. On the other hand, frequencies were reused between clusters, so that two stations belonging to different clusters could perfectly transmit in the same channel. This allows the system to serve more users, at the price of having a certain interference level remaining, which depends on the so called reutilization distance, i.e. the distance between any two cells sharing the same channels.

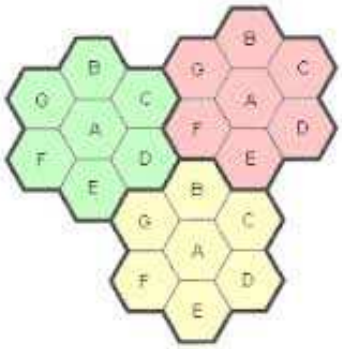
This concept of frequency reuse opened the topic of cluster design and channel allocation as a research line. A complete theory based on cellular geometry, that is, the study of the geometry induced by hexagonal grids, was created. A trade-off appears regarding the cluster size: bigger clusters reduce the interference but use the bandwidth less efficiently, thus serving less users. Moreover, clusters were allowed to have only certain sizes, actually, those that were rhombic numbers (a rhombic number is an integer which can be expressed as $n = i^2 + j^2 + ij$ for any two integers i and j). Examples can be found in figure 5.1.

In the recent years, MIMO base-station coordination has made necessary again to find ways of grouping base-stations and researches worldwide seem to be using the same cluster structures inherited from cellular geometry theory, developed in the era of GSM, without ever wondering whether this new application would be better served with a newer grouping scheme.

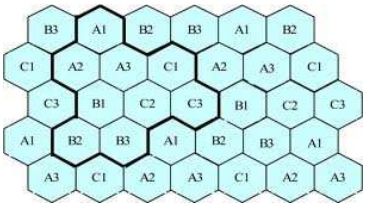
During this project, we will investigate whether adaptively partitioning the set of cells accord-



(a) A cluster with 4 base-stations



(b) A cluster with 7 base-stations



(c) A cluster with 9 base-stations

Figure 5.1: Examples of typical cluster patterns employed in GSM for frequency reuse.

ing to the particular signal propagation conditions at a given time instant provides a significant advantage with respect to static partitions in BTS MIMO coordinated transmission.

To be precise, we can formulate our objective as a constrained, discrete optimization problem: given a set of base-stations arranged according to some arbitrary deployment, we try to find the partition which maximizes some cost function reflecting the system performance, subject to a certain set of constraints on the allowable cluster sizes. We assume that the signal processing techniques are fixed a priori. In other words, given a certain signal processing algorithm to obtain the transmit precoder $\mathbf{W_{tx}}$ and receive filter $\mathbf{W_{rx}}$ in each cluster, the clustering algorithms we develop will try to find the partition which works better for that given MIMO filtering method. Employing different techniques in each cluster could be an interesting future line of research but, for now, we avoided it for the sake of simplicity.

The cluster size constraints account for the feasibility limits imposed by the difficulties involved in the coordination of too many BTSs and the estimation of too many channels.

The simplest case we can consider is to request that all the resulting clusters in the partition have no more than $L_{\max}$ BTSs.

We also consider a three-parameter approach which introduces more flexibility. It defines two cluster size thresholds, $L_{\max,a}$ and $L_{\max,b}$, with $L_{\max,b} > L_{\max,a}$. The idea is that most clusters need to contain at most $L_{\max,a}$ base-stations but a small number of clusters, P, are allowed to have up to $L_{\max,b}$ stations. This scheme was designed to account for the fact that, when automatic clustering algorithms are used, cluster sizes tend to exhibit some variance around the average cluster size. Therefore, if any of those algorithms was to be compared with a scheme with identical clusters like those employed in GSM, setting $L_{\max}$ equal to the cluster size in the fixed scheme would lead to having an average and median cluster size significatively smaller in the machine-learning based partition, making the comparison unfair. Note also that setting $P = 0$ recovers the simpler criterion based on a single threshold.

In any way, in a more realistic setup, it would be up to the mobile communications operator to decide which are the exact cluster size constraints which reflect the implementation costs that it is willing to accept. The structure of the algorithms developed in this project is completely general and modular in such a way that adapting them to deal with more sophisticated sets of maximum cluster size constraints would be a trivial task.

Similarly, there are many possible choices for the cost function, such as the system's median rate, sum-rate, the total mean-square error or the average bit error rate (BER).

The optimization problem we just introduced can be readily seen to be NP-hard. For any practical scenario with a realistic number of base-stations, its complexity is simply too big to think of trying to find the optimal solution.

Nonetheless, this project was born under the hope that applying state-of-the-art clustering algorithms from the field of machine learning, such as the ones introduced in chapter 4, may allow us to obtain useful partitions of the set of base-stations we are dealing with for MIMO coordinated transmission. To understand the relation, imagine that we associate base stations with "observations" of a "virtual" data set, where each BTS is mapped to a 2-D vector given by the X-Y coordinates of its physical location. Besides, by employing information about the channel and/or the corresponding set of user locations, we can construct similarity (dissimilarity) metrics between such "observations".

However, it is important to realize that, even though the problem currently at hand bears many similarities with clustering applications in data analysis, it has also some fundamental differences.

On the one hand, clustering is essentially an unsupervised machine learning technique. As we discussed in chapter 4, the main idea is to group similar objects together, but the concept of similarity had an inherent ambiguity so that different definitions of similarity could lead to many perfectly valid partitions, with no real way of assessing which one was better. However,

in our particular case, we do have very clear, objective measures to determine which partitions are better and which ones are worse. In other words, our application is actually not a clustering problem as it is usually understood in machine learning but, rather, we are using clustering algorithms as a heuristic to obtain reasonably good solutions of an extremely complex, discrete optimization problem.

On the other hand, the existence of constraints in the cluster size is something extremely rare in the field of machine learning. As we discussed in the previous chapter, many clustering algorithms control in an indirect way the granularity of the partitions, which in turn is essentially related to the average cluster size. For instance, some algorithms, such as K-means, fix the resulting number of clusters a priori, which serves as a way of roughly setting the resulting cluster size. However, there is nothing which really ensures that clusters will be balanced, i.e. having roughly the same number of base-stations. As a consequence, all the algorithms we studied in chapter 4 provide no way of ensuring that the maximum cluster size specifications are not violated.

Because of that, we have had to modify the algorithms explained in chapter 4 to fit the particularities of BTS grouping for MIMO coordinated transmission in cellular environments. Each particular algorithm has experienced its own modifications: some of those had the motivation of allowing to establish a hard limit on the maximum allowable cluster size whilst others tried to better reflect the nature of the magnitude to be optimized.

After a great amount of research and empirical evaluations, we ended up employing a generic, four-step structure in all the particular algorithms we have developed, as shown in algorithm 5.1.

This general structure could actually be simplified in some particular cases. For example, most clustering algorithms to be used during step 1 generate partitions with clusters containing a single connected component. This is a direct consequence of the fact that, due to the strong decay of signal with distance, it is very rare that two base-stations in cells which are not direct neighbors interfere each other while not interfering their direct neighbors. However, some algorithms, specially the one based on hierarchical agglomerative clustering, tends to produce residual clusters consisting of many connected components as we get nearer the root of the dendrogram. Besides, this step is completely harmless for the algorithms which do not need it and the computational complexity overhead is moderate, hence we believe that it is interesting to keep it.

Similarly, some of the clustering algorithms we developed for step 1 are designed so that their output partition already satisfies the size constraints. Therefore, when those algorithms are being used, we could avoid the third step.

However, our experiments suggested that, even in those particular cases, the usage of this generic, four-step algorithm opens more degrees of freedom allowing us to increase the algorithm's performance. For instance, in algorithms with a built-in maximum cluster size limitation, we explored setting the size limit of the main clustering algorithms to a certain threshold

Algorithm 5.1: Generic structure for BTS clustering algorithms applied to MIMO coordinated transmission in cellular systems

input : A set of M base-stations, $\mathbb{B} = \{b_1, b_2, \ldots, b_M\}$, characterized by their spatial locations, i.e. their X-Y coordinates $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \mid \mathbf{x}_i \in \mathbb{R}^2$

input : Information about instantaneous signal propagation conditions: MIMO channel matrix $\mathbf{H}$ and/or set of user spatial locations, characterized by their X-Y coordinates $\mathbb{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M\} \mid \mathbf{y}_i \in \mathbb{R}^2$

input : Lower and upper cluster size thresholds, $L_{\max,a}$ and $L_{\max,b}$, together with the number of clusters allowed to reach a size of $L_{\max,b}$ stations, P

output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of the set of base-stations $\mathbb{B}$, such that $|\mathbb{S}_i| \leq L_{\max,b}\ \forall i$ and $\sum_{i=1}^{k} [|\mathbb{S}_i| > L_{\max,a}] \leq P$, with $[\bullet]$ denoting the conditional operator which evaluates to 1 if its argument is true and 0 otherwise

1 Obtain an outline of the partition using any of the clustering algorithms discussed in section 5.2 ;

2 Every cluster containing several connected components is split with each of the connected components originating a new cluster as discussed in section 5.3 ;

3 The partition is iteratively modified by splitting every cluster which violates the cluster size constraints, using the algorithms exposed in section 5.4 ;

4 Sets of small neighboring clusters which could be joined while still satisfying the cluster size constraints are merged with the algorithm of section 5.5 ;

$L'_{\max} > L_{\max}$ and later "correcting" the clusters which actually had more than $L_{\max}$ base-stations during step 3. In this way, we found that if the parameter $L'_{\max}$ was carefully chosen, it was possible to outperform the alternative of employing $L'_{\max} = L_{\max}$ and eliminating step 3.

Because of that, we decided to stick to the pseudo-code shown in algorithm 5.1 and tune the parameters properly for each particular case.

Once we have introduced the general structure of all the base-station clustering algorithms developed in this project, the following sections will explain in greater detail each of the particular steps and the different implementations we have considered.

5.2 Core clustering algorithms

The key step in algorithm 5.1 is the first one, in which we use algorithms inspired in the clustering algorithms of chapter 4 to obtain a rough sketch of the resulting clusters. Even though the subsequent steps will "refine" this initial partition, the changes introduced are relatively minor, so that it is safe to say that the performance of algorithm 5.1 is mainly determined by the quality of the clustering algorithm employed during step 1.

During the remainder of this section, we will explain each of the core clustering algorithms

we have developed for such purpose.

We will comment the modifications we made to the original clustering algorithm in which they are based to adapt it to the needs of our application, BTS grouping for coordinated MIMO transmission; discuss the algorithm's parametrization and, finally, outline the pseudo code.

5.2.1 Similarity functions for base-station coordination in MIMO cellular systems

Before actually discussing the base-station clustering algorithms themselves, we are going to study the design of similarity (dissimilarity) metrics useful in the context of grouping base-stations rather than objects of a certain data set. Those will be a fundamental part of the first three algorithms we have proposed.

It is easy to realize that using the spatial locations of the base-stations alone is not enough to find any edge to be exploited in order to enhance the system's performance. Indeed, if no information at all about the instantaneous signal propagation conditions is available, we fall back to geographical considerations such as those used to create clusters in the times of GSM.

Precisely, considering that interference between cells belonging to the same cluster will fall to negligible levels thanks to the signal processing algorithms coordinating the joint transmission within the cluster, we have that the main limiting factor will be inter-cluster interference. Besides, because path-loss in urban environments causes signal strength to decay quite fast with distance, as a rough approximation, each cell can be assumed to interfere only its immediate neighboring cells. As a consequence, inner cells within a cluster, i.e. those whose neighbors all belong to the cluster, will experience very favorable conditions on average. On the other hand, cells which are at the frontier between several clusters will experience considerable interference levels limiting the maximum achievable rate.

Because of that, whenever only the spatial distribution of the base-stations is known, the optimal way to form clusters is by selecting shapes which, given a fixed cluster size, maximize the number of inner cells within the cluster.

The best possible solution of this kind is obtained by employing hexagonal cluster shapes, that is, clusters consisting of a central cell and T tiers around it. In figures 5.2 and 5.3 we show two examples for $T = 1$ and $T = 2$. It is easy to see that the number of BTSs in a cluster like that is $1 + 6\sum_{n=1}^{T} n = 3T^2 + 3T + 1$, being $6T$ the number of cells in the cluster's boundary. Then, as T increases, the proportion of cells suffering severe interference converges asymptotically to 0 as $2/T$. However, because the number of base-stations is $3T^2 + 3T + 1$, only the solution for $T = 1$ and, for some special cases, $T = 2$, are feasible in practice: that is, this approach is very rigid. Nonetheless, the solution with hexagonal clusters of 7 BTSs, corresponding to $T = 1$, is extensively used up to the point of being one of the most common assumptions done by researchers attempting to study base-station coordination in MIMO environments. For cluster sizes which are not of the form $3T^2 + 3T + 1$, other shapes need to be used but always keeping in mind that the main objective is to have the lowest possible number of cells in the border, an

approach which leads to the usage of GSM-like clusters based on cellular geometry.

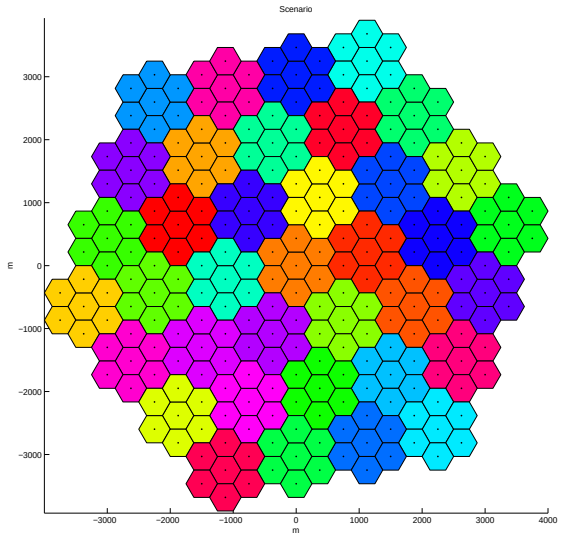


Figure 5.2: $T = 1$: Fixed hexagonal clusters with 7 base-stations.

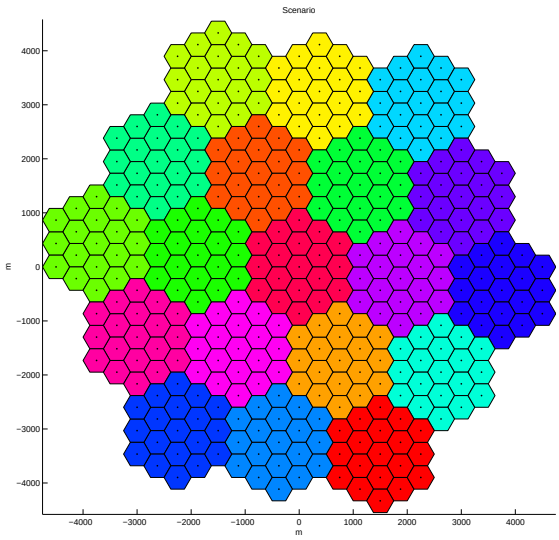


Figure 5.3: $T = 2$: Fixed hexagonal clusters with 19 base-stations.

As we discussed in the introduction, mobile communications have reached such high levels of spectral efficiency that even the slightest improvement can be really interesting. Because of that, in our opinion, it is a major waste to coordinate the base-stations in clusters which were made ignoring all the relevant instantaneous information on the signal propagation conditions.

Cellular systems behave as living things: users are constantly moving from one location to

another, some of them at tens of kilometers per hour, and particular events can originate unusual accumulations of users randomly at any location within the coverage area; the environment in which the signal propagates is time-varying, since weather, pollution and even urban traffic such as cars and buses may introduce attenuation, scattering and reflections which are constantly changing the conditions characterizing the channel; and, last but not least, Rayleigh fading itself can also wildly modify the interference patterns between base-stations affecting the whole system. As a consequence, the signal propagation conditions at a particular instant may be radically different than those just a quarter of hour later.

Then, if we are able to dynamically change the partition, adapting the clusters to the particular conditions affecting the system we can obtain a performance gain which, given the current background and trends in mobile communications, will become a necessary asset in the future.

From an intuitive point of view, since the within-cluster interference will be almost eliminated by coordinated MIMO signal processing, we want that cells which interfere each other significatively are assigned to the same cluster and cells with small mutual interference belong to different clusters. As clustering is designed so that similar objects are clustered together, it follows that, for clustering of base-stations for coordination in MIMO systems, we need to build our similarity functions in such a way that two BTSs are to be considered similar if they mutually interference with each other considerably, and dissimilar otherwise.

Depending on the amount of information available and the way we use it, we have designed up-to four different similarity functions:

Unnormalized interference similarity: In chapter 2, we proved that the interference power received by the user in the i-th cell can be obtained from equation (2.38), where

$$P_{\text{int},j\to i} = \text{Tr}((\widetilde{\mathbf{H}})_{i,j}\mathbf{R}_{\mathbf{u}_j}(\widetilde{\mathbf{H}})_{i,j}^H) \tag{5.1}$$

accounts for the interference power caused by the j-th cell. Under the common assumption that the transmitter generates spatially white symbols, it follows that the interference received by the i-th user due to the transmission originated in the j-th base-station is given by the squared Frobenius norm of $(\widetilde{\mathbf{H}})_{i,j}$, that is, $P_{\text{int},j\to i} = \left\|(\widetilde{\mathbf{H}})_{i,j}\right\|_F^2 = \text{Tr}\left((\widetilde{\mathbf{H}})_{i,j}(\widetilde{\mathbf{H}})_{i,j}^H\right)$. Because in order to obtain the system-wide precoder $\mathbf{W_{tx}}$ and receive filter $\mathbf{W_{rx}}$ we need to establish the clusters first, it is not possible to employ $\left\|(\widetilde{\mathbf{H}})_{i,j}\right\|_F^2$ as a way to quantify the similarity between base-stations i and j.

Nevertheless, we can use the amount of interference that would be generated without coordination as a rough estimation of the desired magnitude, that is, we can define the similarity between cells i and j as the interference power received by the user in the i-th cell due to the transmission originated in the j-th cell in the particular case in which no MIMO signal processing is used at all. This would imply that $\mathbf{W_{tx}} = \mathbf{I}_{Mt\times Nr}$ and $\mathbf{W_{rx}} = \mathbf{I}_{Nr\times Nr}$, hence, $\widetilde{\mathbf{H}} = \mathbf{H}$. Therefore, the similarity could then be defined as:

$$w'_{ij} = \|(\mathbf{H})_{i,j}\|_F^2 \tag{5.2}$$

The similarity metric defined in equation (5.2) is valid and makes perfect sense given the context of our application yet it is not fully satisfactory, which explains why we introduced the notation with primed similarities w'_{ij}. The flaw of the previous definition is that it is not reciprocal, that is, $w_{ij} \neq w_{ji}$. As far as spectral clustering is concerned, this would imply that the graph representation of our scenario would be a directed weighted graph, whereas we are interested in weighted graphs which are undirected.

In order to introduce symmetry into the adjacency function expressed in equation (5.2), we redefine the concept by saying that the adjacency between cells i and j will be measured as the average of the interference power received by the i-th user due to the transmission originated in the j-th base-station and the interference power received by the j-th user due to the transmission originated in the i-th base-station, both under the assumption that interference powers are evaluated before signal processing, that is, $\widetilde{\mathbf{H}} = \mathbf{H}$. Mathematically:

$$w_{ij} = w_{ji} = \frac{1}{2}\left(\|(\mathbf{H})_{i,j}\|_F^2 + \|(\mathbf{H})_{j,i}\|_F^2\right) \tag{5.3}$$

Note that an interesting alternative to the decision of computing interference powers before signal processing would be to measure them considering an scenario with no BTS coordination, i.e. considering that each cell obtains its own precoder $\mathbf{W_{tx}} \in \mathbb{C}^{t\times r}$ and receive filter $\mathbf{W_{rx}} \in \mathbb{C}^{r\times r}$ independently of the other cells, which in turn is the same as considering a partition formed by M singleton clusters. However, the main drawback of this approach is the increase of computational complexity due to the necessity of obtaining M precoders with their corresponding M received filters only to obtain the similarity matrix $\mathbf{W}$ of the graph.

Normalized interference similarity: Due to the randomness inherent to the user location and Rayleigh fading, the desired signal power may vary quite a lot from cell to cell. Based on that observation, we believed then that it was worth trying a normalized version of the similarity function defined in equation (5.3), in which the interference power received by the i-th user is normalized by the desired signal power at that user's receiver. Mathematically:

$$w_{ij} = w_{ji} = \frac{1}{2}\left(\frac{\|(\mathbf{H})_{i,j}\|_F^2}{\|(\mathbf{H})_{i,i}\|_F^2} + \frac{\|(\mathbf{H})_{j,i}\|_F^2}{\|(\mathbf{H})_{j,j}\|_F^2}\right) \tag{5.4}$$

BTS-user distance similarity: As we discussed, in practice, the estimation of the complete channel matrix $\mathbf{H}$ is unfeasible.

One perfectly valid way to deal with that problem is to implement a channel estimation procedure in which each BTS estimates the channels to users in cells belonging to the first or second tier around its own cell. The rest of users would be considered to be too far, so

that the path loss is taken to be infinite, implying $(\mathbf{H})_{i,j} = \mathbf{0}$ for all users i not located in cells belonging to the first or second tier around the j-th cell.

The high amount of path-loss present in urban environments justifies the previous approximation. However, it is also interesting to consider simpler schemes which use only information about the user's locations. Another important advantage of this approach is that, apart from the saving in computational complexity, users move much slower than the system channel matrix $\mathbf{H}$ changes. In other words, algorithms based on the location of the users rather than on the channel matrix $\mathbf{H}$ require a significatively lower refresh rate to keep track of changes in the signal propagation conditions.

The first scheme we propose to employ the user locations as a way of defining similarities between cells is very simple.

Let $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \mid \mathbf{x}_i \in \mathbb{R}^2$ be the set of base-station locations, expressed as their spatial coordinates. Similarly, let $\mathbb{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M\} \mid \mathbf{y}_i \in \mathbb{R}^2$ be the corresponding set of user locations, in such a way that $\mathbf{y}_i$ contains the spatial coordinates of the user served by the base-station located at $\mathbf{x}_i$.

We begin by assuming, as it is indeed the case, that path-loss is the dominant effect characterizing signal propagation and thus the amount of interference. Then, we can consider that the smaller the distance between the i-th user and the j-th base station, $\|\mathbf{y}_i - \mathbf{x}_j\|$, the bigger the interference affecting the i-th user's receiver due to the transmission originated in the j-th cell. In other words, $\|\mathbf{y}_i - \mathbf{x}_j\|$ is a reasonably good dissimilarity metric, which can be converted into a similarity metric by applying a non-decreasing mapping as we discussed in chapter 4.

Given that signal strength is usually modeled to decrease with the distance R as $R^{-\gamma}$, an exponential decay of similarity with distance seems a rather appropriate choice. Moreover, considering that base-stations are equipped with isotropic antennas, a radially-symmetric Gaussian kernel can be employed to transform the dissimilarities $\|\mathbf{y}_i - \mathbf{x}_j\|$ into a similarity metric as:

$$w'_{ij} = e^{\frac{-\left\|\mathbf{y}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}} \tag{5.5}$$

Nonetheless, because the user locations are completely random, with probability close to 1 we have that $\|\mathbf{y}_i - \mathbf{x}_j\| \neq \|\mathbf{y}_j - \mathbf{x}_i\|$ and, as a consequence, w'_{ij} as in equation (5.5) also defines a directed weighted graph. Using the same idea as before, we can define a symmetric similarity metric by averaging the weights for both senses of the graph as:

$$w_{ij} = w_{ji} = \frac{1}{2}\left(e^{\frac{-\left\|\mathbf{y}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}} + e^{\frac{-\left\|\mathbf{y}_j - \mathbf{x}_i\right\|^2}{2\sigma^2}}\right) \tag{5.6}$$

An interesting idea is that we can straightforwardly extend our work to cover the usage of directive antennas by employing a complete covariance matrix in the expression of the Gaussian kernel. That is,

$$w'_{ij} = e^{-\frac{1}{2}(\mathbf{y}_i - \mathbf{x}_j)^T \boldsymbol{\Sigma}^{-1} (\mathbf{y}_i - \mathbf{x}_j)} \tag{5.7}$$

In this particular case, $\boldsymbol{\Sigma}$ would be constructed from its eigenvectors and eigenvalues so that the principal axis would point towards the direction in which the directivity is maximum and the eccentricity would be proportional to the gain of the antenna.

User distance similarity: From an implementation point of view, the last similarity metric we propose resembles the previous one a lot. Actually, it is even simpler: we measure the dissimilarity between two cells i and j as the distance between the users being served by those cells.

Even though this scheme does not strictly represent interference, it benefits from some other advantages. On the one hand, this metric is inherently symmetric, so that there is no need to include any external averaging process as we had to do for the other three similarity functions we proposed. Also, this metric is more suitable to detect anomalous accumulation of users around certain locations, which is actually a base-station clustering criterion as powerful as any of the interference-based ones.

Mathematically, we can then express the user-distance based similarity as:

$$w_{ij} = w_{ji} = e^{\frac{-\left\|\mathbf{y}_i - \mathbf{y}_j\right\|^2}{2\sigma^2}} \tag{5.8}$$

Where radially-symmetric Gaussian kernels have been employed for the same reasons we previously argued. Note that this similarity function can also be adapted to deal with directive antennas.

As a final consideration, note that we have not defined any kind of similarity which only takes into account the distance between base-stations. The reason for that is actually pretty simple: if we want our algorithms to be dynamic in the sense that they are able to adapt to the instantaneous signal propagation conditions, we cannot employ a similarity function which is based on magnitudes which are constant over time. Besides, in regular deployments such as the ones we work with, the set of spatial locations of the BTSs contains no discriminative information relevant for designing the clusters.

To conclude our discussion on the design of similarity functions for base-station clustering in MIMO coordinated transmission, we provide as an example graphical representations of the similarity matrices $\mathbf{W}$ obtained using each of the four methods we proposed for the scenario shown in figure 5.4. In figure 5.5, the brightness of each square (i, j) is proportional to the

similarity $(\mathbf{W})^{i,j}$. The maximum similarity value in each matrix, $\max_{i,j}(\mathbf{W})^{i,j}$, is assigned white color whereas the rest of cells are colored in a 8-bit grayscale in such a way that darker colors correspond to lower similarities. Because the range covered by the similarities in natural "units" is too big, the coloring is actually assigned proportionally to the logarithm of the similarities. Otherwise, the images would all look mostly black with some white dots scattered around.

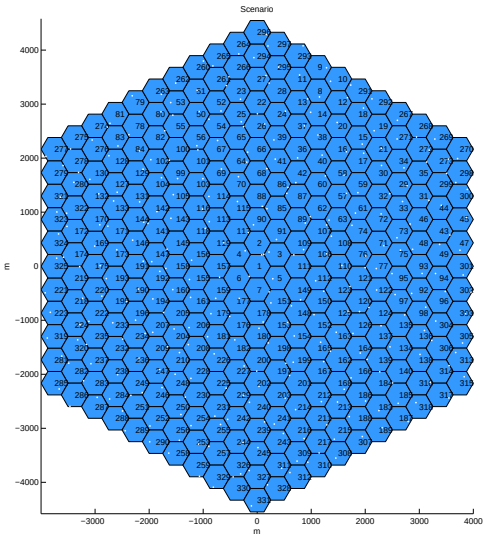


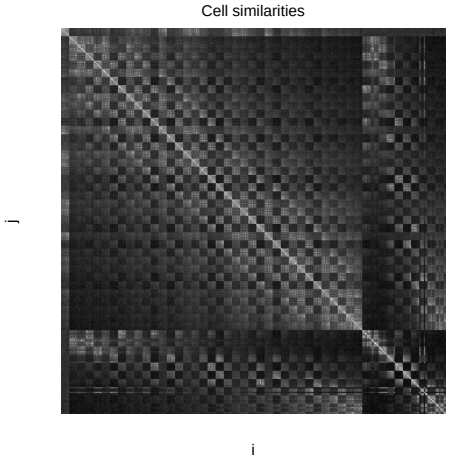
Figure 5.4: Example of a BTS deployment. White dots represent user locations.

5.2.2 Spectral base-station clustering

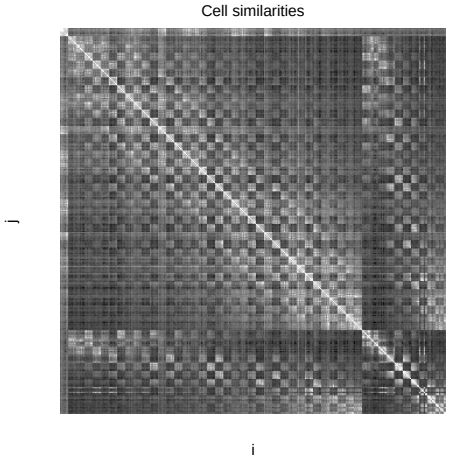
The first algorithm we devised is a straightforward application of spectral clustering as discussed in section 4.4. Actually, the algorithm itself can be chosen from any of the spectral clustering variants discussed in section 4.4.5 without including a single modification in the pseudo-code. The only yet fundamental difference with respect to general purpose spectral clustering lies in the similarity metrics previously discussed.

5.2.2.1 Pseudo-code

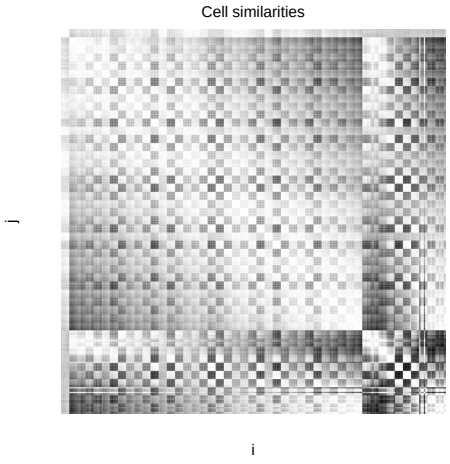
As we can see in algorithm 5.2 table, its pseudo-code is ridiculously simple: we simply invoke any of the spectral clustering routines discussed in chapter 4 but using a context-sensitive similarity matrix $\mathbf{W}$, which is in this case the key allowing to obtain good solutions when grouping base-stations for coordinated MIMO transmission out of a family of general purpose clustering algorithms.



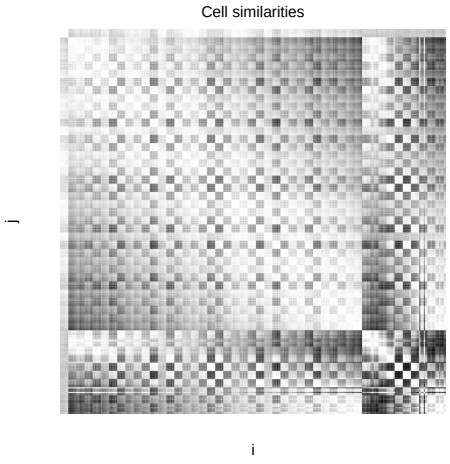
(a) Unnormalized interference similarity



(b) Normalized interference similarity



(c) BTS-user distance similarity with $\sigma = 2R_{\text{cell}}$



(d) User distance similarity with $\sigma = 2R_{\text{cell}}$

Figure 5.5: Color plot of $\mathbf{W}$ *as obtained using each of the four similarity functions described in this section for the scenario depicted in figure 5.4.*

5.2.2.2 Parametrization

Algorithm 5.2 contains very few external parameters which can be modified for adjusting its behavior. Depending on the chosen similarity function, we only have access to one or, at most, two parameters.

One of the fundamental parameters, which is always available regardless on our choice about the similarity metric, is the desired number of clusters k, directly inherited from the family of spectral clustering algorithms. It can be used for coarse tuning of the average cluster size: if we want to partition a set of M base-stations in clusters of $L_{\max}$ cells, the expected number of clusters should be in the order of $k \approx \text{round}(\frac{M}{L_{\max}})$.

Algorithm 5.2: Spectral base-station clustering algorithm

input : A symmetric similarity matrix $\mathbf{W}$ representing the set of base-stations $\mathbb{B} = \{b_1, b_2, \ldots, b_M\}$, obtained using one of the similarity functions described in section 5.2.1

input : Desired number of clusters, k

output: A coarse partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of the set of base-stations $\mathbb{B}$, to be employed within algorithm 5.1 as a basis for the subsequent, partition-refining steps

1 Apply any of the spectral clustering algorithms described in section 4.4.5 with a desired number of clusters k and using $\mathbf{W}$ as the weighted adjacency matrix for the graph associated to the clustering problem to obtain the output partition $\mathbb{S}$;

Spectral clustering penalizes partitions with unbalanced clusters smoothly because of the family of implicit cost functions, ratio and normalized cuts, which they attempt to optimize under the spectral relaxation approach. However, precisely because the penalization is smooth, we have no guarantee that the maximum cluster size constraints will be satisfied at all. Therefore, the parameter k alone cannot be used in our application to fulfill the feasibility specifications.

Besides, as we previously discussed, the general structure for base-station clustering shown in algorithm 5.1 already contains an artifact to ensure constraint satisfaction in the third step. Therefore, in the end, the desired number of clusters k is a parameter which we employ as an extra degree of freedom to be tuned for optimal performance.

Apart from k, when BTS-user distance or user distance based similarities are employed, the bandwidth or standard deviation of the radially symmetric kernel, σ, is the other main parameter which, by fine tuning the decay of the similarity with distance, becomes a fundamental tool for optimizing the algorithm's performance.

5.2.3 Hierarchical divisive spectral base-station clustering

The algorithm we will discuss during this section was originated as the most natural extension of algorithm 5.2 when trying to further adapt the family of spectral clustering algorithms, intended for data-analysis applications, to our particular problem in the field of next generation mobile communications. To be completely honest, the main motivation for the development of hierarchical divisive spectral base-station clustering came at a stage of this project in which we had not yet devised the general structure shown as algorithm 5.1 and, thus, we were still struggling to find ways of introducing the maximum cluster size constraints into standard, general purpose clustering algorithms like those of chapter 4.

Because substituting the smooth penalization for unbalanced clusters contained in spectral clustering algorithms by any type of hard constraint on the maximum cluster size seems an unsurmountable task, given that the cost function would have to be terribly distorted into some

kind of non-differentiable function, our first idea by then was to mix spectral clustering with hierarchical clustering. In this way, it would be easier for us to control the size of the resulting clusters by iteratively dividing them until all cluster sizes were below the desired threshold.

5.2.3.1 Pseudo-code

Algorithm 5.3: Hierarchical divisive spectral base-station clustering algorithm

input : A symmetric similarity matrix $\mathbf{W}$ representing the set of base-stations $\mathbb{B} = \{b_1, b_2, \ldots, b_M\}$, obtained using one of the similarity functions described in section 5.2.1

input : A maximum cluster size, $L_{\max}$

output: A coarse partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of the set of base-stations $\mathbb{B}$, to be employed within algorithm 5.1 as a basis for the subsequent, partition-refining steps

```
Initialize the partition 𝕊 by creating a unique cluster 𝕊_1 which contains the all the
base-stations, 𝕊_1 = 𝔹 ;
while ∃ 𝕊_j ∈ 𝕊 : |𝕊_j| > L_max do
    for j : |𝕊_j| > L_max do
        Split the cluster 𝕊_j in two by applying any of the spectral clustering algorithms
        described in section 4.4.5, with a desired number of clusters k = 2, and using the
        portion of W relevant for cells belonging to cluster 𝕊_j as the weighted adjacency
        matrix for the graph associated to the clustering problem ;
    end
end
The partition 𝕊 already satisfies the considered maximum cluster size constraint and is
the algorithm's output ;
```

The idea in which the pseudo-code outlined in algorithm 5.3 is based could not be any simpler.

Just as algorithm 5.2, we use spectral clustering with the similarity metrics we introduced in section 5.2.1 to account for the particularities of our peculiar scenario. However, rather than using a spectral clustering algorithm just once with the complete similarity matrix $\mathbf{W}$ to directly find the partition, we use a divisive hierarchical procedure which has a stopping criterion depending on the size constraint.

In this way, the partition $\mathbb{S}$ is initialized with a single macro-cluster containing all the base-stations. After that, spectral clustering is iteratively applied to split any cluster in the current partition $\mathbb{S}$ which contains more than $L_{\max}$ cells in two clusters. The algorithm will stop when all the clusters have at most $L_{\max}$ cells, hence satisfying the maximum cluster size constraint.

Therefore, algorithm 5.3 is actually following the same pseudo-code as any general purpose

hierarchical divisive clustering algorithm: all the observations are assigned initially to the same cluster and, iteration after iteration, the next partition in the hierarchy is obtained by splitting one or more clusters of the current partition into two or more sub-clusters, employing whatever clustering algorithm we want to implement the splitting process.

However, there is a fundamental difference between algorithm 5.3 and typical hierarchical divisive clustering algorithms such as 4.3. In data-analysis, the biggest attractive of hierarchical clustering, be it divisive or agglomerative, is precisely its capacity to produce not a single partition but a complete hierarchy of partitions at different granularity levels. However, in our application, we are not interested in that. Instead, we want to use the iterative procedure of hierarchical clustering to have some way of limiting the cluster size. Because of that, every new partition replaces the previous, that is, partitions of intermediate iterations are discarded. Even more importantly, because we are not interested in obtaining the complete hierarchy, we do not continue the cluster division process until all clusters are singletons. Rather, we are content once a partition with all its clusters satisfying the maximum cluster size constraint is found. As a consequence, even though we included the work "hierarchical" in the name of the algorithm because it was inspired in hierarchical divisive clustering, the resulting pseudo-code is probably closer to being recursive than hierarchical.

As a side note, despite the fact that in hierarchical divisive spectral base-station clustering we have only included a simple, one-threshold cluster size constraint, it would be very easy to redefine the algorithm to deal with more exotic specifications. In the end, the only thing we would need to do is, in each iteration, look for the clusters which are not fulfilling whatever size constraints we are considering and split them into two using spectral clustering.

5.2.3.2 Parametrization

The key player in the algorithm's performance is still the similarity function. All the possible choices discussed for algorithm 5.2 still apply to algorithm 5.3. Moreover, the parametrization conditions are the same: the two interference-based similarity functions have no parameters whereas the two similarity functions based on distances have the parameter σ for further tuning.

However, the introduction of the divisive approach substitutes the desired number of clusters k by the maximum cluster size $L_{\max}$.

Initially, our intention was to use $L_{\max}$ not as a parameter but as a way of introducing a maximum cluster size constraint into clustering algorithms. However, after we devised the general structure for base-station clustering shown in algorithm 5.1, we empirically checked that it was more interesting to let the third step take care of ensuring the satisfaction of the maximum cluster size constraints and use $L_{\max}$ in algorithm 5.3 as an extra parameter to be optimized for extra performance.

5.2.4 Hierarchical agglomerative base-station clustering

Hierarchical agglomerative base-station clustering bears a strong relation with 5.3. We followed the same concepts, i.e. applying hierarchical clustering to enforce a maximum cluster size constraint. However, now we apply agglomerative clustering instead of divisive clustering. Therefore, the resulting algorithm is somewhat the opposite of algorithm previously discussed.

On the one hand, hierarchical agglomerative clustering does not use another clustering algorithm in any step: it is an algorithm driven only by the pairwise similarities contained in the similarity matrix $\mathbf{W}$. That's the reason why the adjective spectral is no longer present in the algorithm's denomination.

On the other hand, hierarchical agglomerative clustering begins by considering each cell to be a cluster and iteratively merges clusters depending on their similarities. However, since in our application we do not want to obtain the complete hierarchy of partitions, we will not continue the merging process until all the cells belong to the same cluster but, rather, we will stop the process whenever merging any two clusters would create one with more than $L_{\max}$ cells.

5.2.4.1 Pseudo-code

The pseudo-code of algorithm 5.4 is actually just a slight modification of the basic agglomerative hierarchical clustering algorithm 4.2 we saw in chapter 4.

We begin by initializing the partition $\mathbb{S}$ assigning each cell to a different cluster. This results on a partition of M singletons which is equivalent to an scenario with no coordination between base-stations for MIMO signal processing. In the next step, we initialize a collection $\mathbb{I}$ of the clusters "eligible" for merging, that is, those which contain less than $L_{\max}$ cells. Note that, as we merge clusters, the cardinality of $\mathbb{I}$ will reduce progressively.

Then, while there are "eligible" clusters remaining, we obtain the cluster assignation matrix $\mathbf{E}$ associated to the current partition $\mathbb{S}$ and use it to compute the similarities between clusters as $\mathbf{W}_{\mathbb{S}} = \mathbf{E}^T\mathbf{W}\mathbf{E}$. With this notation, $(\mathbf{W}_{\mathbb{S}})^{i,j}$ is the similarity between the i-th and j-th clusters. The reader familiar with the concept of linkages for agglomerative clustering may be a bit surprised, because the linkage we just defined is very similar to average linkage but the normalization by the respective cluster sizes seems to be missing. That is, average linkage as it was defined in chapter 4 is formulated as:

$$(\mathbf{W})^{\mathbb{S}_i,\mathbb{S}_j} = \frac{1}{|\mathbb{S}_i||\mathbb{S}_j|} \sum_{k:\mathbf{x}_k\in\mathbb{S}_i} \sum_{l:\mathbf{x}_l\in\mathbb{S}_j} (\mathbf{W})^{k,l} = \frac{1}{|\mathbb{S}_i||\mathbb{S}_j|}(\mathbf{E}^T\mathbf{W}\mathbf{E})^{i,j} \tag{5.9}$$

Whereas our calculation skips the division by $|\mathbb{S}_i||\mathbb{S}_j|$.

However, there is a reason for that. In our application, similarities are somehow identified with interference, and we want to join the clusters which suffer the biggest total amount of mutual interference. In other words, the most natural approach to the problem is not any of the typical linkages used in agglomerative clustering. Rather, something which we may call *sum-linkage*, in which the similarity between two clusters is measured as the sum of the pairwise

Algorithm 5.4: Hierarchical agglomerative base-station clustering algorithm

input : A symmetric adjacency matrix $\mathbf{W}$ representing the set of base-stations $\mathbb{B} = \{b_1, b_2, \ldots, b_M\}$, obtained using one of the similarity functions described in section 5.2.1

input : A maximum cluster size, $L_{\max}$

output: A coarse partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of the set of base-stations $\mathbb{B}$, to be employed within algorithm 5.1 as a basis for the subsequent, partition-refining steps

Initialize the partition $\mathbb{S}$ with each cell being a singleton, $\mathbb{S} \leftarrow \{\{b_1\}, \{b_2\}, \ldots, \{b_M\}\}$;
$\mathbb{I} \leftarrow \{1, 2, \ldots, M\}$;
while $\mathbb{I} \neq \emptyset$ **do**
    Obtain cluster assignation matrix $\mathbf{E}$ associated to partition $\mathbb{S}$ of the set of base-stations $\mathbb{B}$;
    $\mathbf{W}_{\mathbb{S}} \leftarrow \mathbf{E}^T \mathbf{W} \mathbf{E}$;
    $\mathbf{d} \leftarrow \mathbf{W}_{\mathbb{S}} \mathbf{1} - \mathrm{diag}(\mathbf{W}_{\mathbb{S}})$;
    $i^* \leftarrow \max_{i \in \mathbb{I}} \mathbf{d}^i$;
    $\mathbb{J} \leftarrow \{j : |\mathbb{S}_{i^*}| + |\mathbb{S}_j| \leq L_{\max}\}$;
    if $\mathbb{J} = \emptyset$ **then**
        $\mathbb{I} \leftarrow \mathbb{I} \backslash i^*$;
    else
        $j^* \leftarrow \max_{j \in \mathbb{J}} (\mathbf{W}_{\mathbb{S}})^{i^*, j}$;
        $\mathbb{S} \leftarrow (\mathbb{S} \backslash \{\mathbb{S}_{i^*}, \mathbb{S}_{j^*}\}) \cup \{\mathbb{S}_{i^*} \cup \mathbb{S}_{j^*}\}$;
        $\mathbb{I} \leftarrow \mathbb{I} \backslash \{i^*, j^*\}$;
        if $|\mathbb{S}_{i^*} \cup \mathbb{S}_{j^*}| < L_{\max}$ **then**
            Append the index associated to the recently created cluster $\mathbb{S}_{i^*} \cup \mathbb{S}_{j^*}$ to collection $\mathbb{I}$;
        end
    end
end
It is not possible to merge any more clusters of partition $\mathbb{S}$ so that such partition is the final output of the algorithm;

similarities between points in each cluster, seems more appropriate. Needless to say, sum-linkage would be a terrible choice for data-analysis applications for many different reasons.

In any way, if the reader wanted to investigate the effects of changing algorithm 5.4 to work with average-linkage or any other linkage studied in section 4.3.1, he/she could implement it just by changing the way $\mathbf{W}_{\mathbb{S}}$ is computed.

After that, we search for the cluster $\mathbb{S}_{i^*}$ which has the biggest sum of "similarities", hence the biggest interference, to all the other clusters. In our context, this is equivalent to looking for the cluster which is suffering and/or causing the biggest amount of interference in the system. A key observation is that the search occurs only over the set of "eligible" clusters, $\mathbb{I}$. Note that without this condition, the algorithm would get inexorably trapped into an infinite loop.

Then, we search for the set of clusters $\mathbb{J}$ which can be merged with cluster $\mathbb{S}_{i^*}$. That is, those clusters which contain at most $L_{\max} - |\mathbb{S}_{i^*}|$ cells.

If there is not a single cluster which can be merged with $\mathbb{S}_{i^*}$, we mark it as "non-eligible" by removing i^* from $\mathbb{I}$. This step is also fundamental to ensure a proper termination of the algorithm.

On the other hand, if $\mathbb{J}$ is non-empty, we look for the cluster $\mathbb{S}_{j^*}$ in $\mathbb{J}$ which has the biggest mutual interference with cluster $\mathbb{S}_{i^*}$. Subsequently, the partition $\mathbb{S}$ is updated by merging clusters $\mathbb{S}_{i^*}$ and $\mathbb{S}_{j^*}$ together. To keep track of the changes, the indexes i^*, j^* of the old, now non-existent clusters $\mathbb{S}_{i^*}$, $\mathbb{S}_{j^*}$ are removed from the collection $\mathbb{I}$ and the index of the newly formed cluster $\mathbb{S}_{i^*} \cup \mathbb{S}_{j^*}$ is added to $\mathbb{I}$ if and only if the resulting cluster is still "eligible", that is, if its size if strictly smaller than $L_{\max}$.

This process is iterated until the list of "eligible" clusters $\mathbb{I}$ becomes empty. When that happens, the current partition $\mathbb{S}$ is the final output of the algorithm.

5.2.4.2 Parametrization

Even though algorithms 5.3 and 5.4 are fairly different, even opposite in a way, the parametrization is identical for both. Because of that, we refer the reader to section 5.2.3.2 for further details.

5.2.5 Mean-shift base-station clustering

In machine learning, spectral clustering is probably one of the most popular clustering techniques nowadays. Not only it provides results which outperform alternative algorithms in many scenarios but it is also interesting from a theoretical point of view due to the existence of many connections to other algorithms in various fields of machine learning.

On the other hand, mean-shift clustering is barely a minor machine learning algorithm which has received little attention, apart from a few selected articles which attempted to prove its usefulness for certain specialized applications such as computer vision.

As we discussed in section 4.5 of the previous chapter, mean-shift is actually an extremely elegant way of jointly implementing a non-parametric PDF estimation based on Parzen windows

followed by a gradient ascent mode-seeking procedure. To put it bluntly, it processes the data with non-parametric methods to estimate the modes of the underlying PDF distribution, that is, the locations in the sample space where we can expect to obtain a higher amount of observations.

To understand why mean-shift adapts so well to our needs, let us consider the following situation.

Imagine that at time instant t_0, the users are scattered more or less uniformly over the coverage area and the clusters for base-station coordination have already been defined according to some unknown procedure. Figure 5.4 could be a good illustration of an scenario like that. However, after an unknown amount of time Δt, some celebrity steps out of the hotel he/she was staying at, located in the coverage area of cell 4. As the paparazzi that gather around the entrance start taking pictures, the scene begins to attract the attention of the people that were passing by. When they realize what is happening, many will decide to take their own photographs and/or videos to capture the moment. Moreover, given the current trends, most of them will also attempt to post the multimedia information they just acquired on different social networks, generating an anomalous outburst of traffic in cell 4 and, as the rumor spreads, in the surrounding cells 1, 2, 6, 117, 119, and 159. However, from an engineering point of view, that situation can have catastrophic results in the infrastructure since, except cell 1, all the other cells have at least 3 neighboring base-stations belonging to another cluster. To put it simply, the users located in those cells which try to transmit big amounts of data simultaneously will interfere with each other so much that they will cause a sudden, pronounced drop in the system's sum-rate and median-rate.

Nonetheless, the real cause of the problem is not the unusual situation but, rather, the poor choice for the clusters. If we had defined one of cluster around cell 4 instead of around cell 1, all the cells involved would have been coordinated and the system would have worked without flaw. Therefore, it would be fantastic if we had some kind of tool that, given the current location of the users, could predict which places are currently having a major user density in order to create clusters around those locations. Luckily for us, that's exactly what mean-shift does.

More importantly, the situation we just described is an "extreme", exaggerated example to make our point clear. However, this kind of scheme is still really beneficial for normal operation of the system in everyday situations. This is because, even though we started by stating that "the users are scattered more or less uniformly over the coverage area", precisely because the user locations are somewhat random, at a given time there will be regions with a high user density and others with fewer users to be serviced. For example, in figure 5.4, we can see that the users in cells 1, 2 and 3 are very close to each other, just like those in cells 71, 73 and 75. In that case, it would be interesting to have cells 1, 2 and 3 assigned to the same cluster $\mathbb{S}_i$, and cells 71, 73 and 75 belonging together to some other cluster $\mathbb{S}_j$. Also, we have to consider that user accumulations in the coverage area have a strongly time-varying nature. For instance, at rush hour many users will be located around transport infrastructures such as train, subway or bus stations, but at Friday night, the areas in town with more night-life will offer a bigger data

traffic. Because of that, it is needed that the clustering algorithm is constantly keeping track of the situation by reestimating which locations have a high user density at each moment.

From an implementation point of view, we will modify the basic mean-shift based clustering algorithms proposed in the literature by taking advantage of one property which we already anticipated in section 4.5. Because it is perfectly possible to carry out the implicit PDF estimation steps in the mean-shift algorithm with a certain data set $\mathbb{Y}$ but still use points in a different data set $\mathbb{X}$ as initial locations for the gradient ascend procedure, we can produce an algorithm which mimics what we believe to be the most natural approach for creating clusters in our application.

In this way, we employ the user locations $\mathbf{y}_i$ to produce a PDF estimate which, in our case, behaves more like a kind of "instantaneous user density evaluation". After that, we use each of the BTS locations $\mathbf{x}_i$ as a seed for a gradient ascent procedure which will end up converging to some of the places in the coverage area which have many users around. Intuitively, it is as if we let each base-station "fall" towards the closest region exhibiting a peak in the user density. Therefore, the set of all base-stations "falling" to the same place naturally define a certain cluster.

5.2.5.1 Pseudo-code

Note that algorithm 5.5 is basically the same as algorithm 4.13, where the set of user locations $\mathbb{Y}$ is used to carry out the PDF estimation and the set of base-station locations $\mathbb{X}$ is set as seeds for the gradient ascent steps. Because of this, we refer the reader to section 4.5 for a detailed explanation of the pseudo-code of algorithm 5.5. Also, it is important to point out that in this context, the set of bandwidth matrices $\{\mathbf{H}_i\}_{i=1}^{M}$ is not to be confused with the channel matrix $\mathbf{H}$ nor any of its blocks $(\mathbf{H})_{i,j}$.

As an example of the operation of this algorithm in our context, we applied algorithm 5.5 to the scenario shown in figure 5.6.

The kernel profile $k(x)$ was selected to be the exponential kernel profile, hence implying that bivariate normal kernels are used for the PDF estimation step.

The bandwidth matrices were obtained using a per-cell scalar bandwidth approach, that is, each user employs a radially symmetric bivariate normal kernel with a different scalar bandwidth h_i. Besides, we computed each of the h_i as the square root of the average squared distance to the 7 nearest neighbors, multiplied by a scalar $\alpha = 0.467$, obtained by linear search. Both the number of nearest neighbors and the scalar parameter α were tuned to optimize the system median rate. In case it is not clear yet, in section 5.2.5.2, we discuss in greater detail different ways to choose the bandwidth matrices $\{\mathbf{H}_i\}_{i=1}^{M}$.

The resulting partition depicted in figure 5.10 was actually obtained using mean-shift base-station clustering in step 1 of algorithm 5.1. In order to provide a fair comparison in terms of average and median cluster sizes with the reference scheme based in fixed 7 BTSs hexagonal clusters as those shown in figure 5.2, we set the lower maximum cluster size to $L_{\max,a} = 7$, the upper maximum cluster size to $L_{\max,b} = 10$ and the number of clusters allowed to surpass the

Algorithm 5.5: Mean-shift base-station clustering algorithm.

input : Set of base-station locations $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \mid \mathbf{x}_i \in \mathbb{R}^2$ and corresponding user locations $\mathbb{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M\} \mid \mathbf{y}_i \in \mathbb{R}^2$

input : Kernel profile $k(x)$

input : Set of bandwidth matrices $\{\mathbf{H}_i\}_{i=1}^{M}$ with $\mathbf{H}_i \in \mathbb{R}^{2\times 2}$

output: A coarse partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_{N_m}\}$ of the set of base-stations $\mathbb{B}$

$p \leftarrow 0$, $\mathbb{M} \leftarrow \emptyset$;

$\mathbf{m}(\mathbf{x}) = \left(\sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\|\mathbf{x} - \mathbf{y}_i\|^2_{\mathbf{H}_i}\right) \mathbf{H}_i^{-1}\right)^{-1} \sum_{i=1}^{N} |\mathbf{H}_i|^{-1/2} g\left(\|\mathbf{x} - \mathbf{y}_i\|^2_{\mathbf{H}_i}\right) \mathbf{H}_i^{-1}\mathbf{y}_i$;

for $j \leftarrow 1$ **to** M **do**

  $t \leftarrow 0$, $\mathbf{x}^{(1)} \leftarrow \mathbf{x}_j$;

  while *convergence criterion not satisfied* **do**

    $t \leftarrow t + 1$, $\mathbf{x}^{(t+1)} \leftarrow \mathbf{m}\left(\mathbf{x}^{(t)}\right)$;

  end

  $\mathbf{x}^{(c)} \leftarrow \mathbf{x}^{(t)}$;

  Perturb $\mathbf{x}^{(c)}$ as $\mathbf{x}_\epsilon = \mathbf{x}^{(c)} + \left\|\mathbf{x}^{(c)}\right\| \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon}$ being a random vector drawn from a bivariate Gaussian distribution with small variance, in the order of 0.01 to 0.1 ;

  $t \leftarrow 0$, $\mathbf{x}^{(1)} \leftarrow \mathbf{x}_\epsilon$;

  while *convergence criterion not satisfied* **do**

    $t \leftarrow t + 1$, $\mathbf{x}^{(t+1)} \leftarrow \mathbf{m}\left(\mathbf{x}^{(t)}\right)$;

  end

  if $\left\|\mathbf{x}^{(t)} - \mathbf{x}^{(c)}\right\| < \text{tol}$ **then**

    $\mathbf{x}_p^{(c)} \leftarrow \mathbf{x}^{(c)}$;

    $p \leftarrow p + 1$, $\mathbb{M} \leftarrow \mathbb{M} \cup \mathbf{x}_p^{(c)}$;

    Associate point $\mathbf{x}_j$ with mode candidate $\mathbf{x}_p^{(c)}$: $C(j) = p$;

  else

    Discard point $\mathbf{x}^{(c)}$ since it corresponds to a saddle point ;

    Annotate that point $\mathbf{x}_j$ has no associated mode candidate for now: $C(j) = -1$;

  end

end

Partition $\mathbb{M}$ in subsets $\mathbb{M}_i = \left\{\mathbf{x}_j^{(c)} \mid \exists\, \mathbf{x}_k^{(c)} \in \mathbb{M}_i \mid \left\|\mathbf{x}_j^{(c)} - \mathbf{x}_k^{(c)}\right\| \leq \text{tol}_h\right\}$. That is, a ball of radius tol_h centered at any point of $\mathbb{M}_i$ must contain at least another point of $\mathbb{M}_i$;

Each of the subset $\mathbb{M}_1, \mathbb{M}_2, \ldots, \mathbb{M}_{N_m}$ defines a distinct cluster ;

for $j \leftarrow 1$ **to** N_m **do**

  $\mathbf{m}_j \leftarrow \frac{1}{|\mathbb{M}_i|} \sum_{\left\{j \mid \mathbf{x}_j^{(c)} \in \mathbb{M}_i\right\}} \mathbf{x}_j^{(c)}$

end

$\mathbb{S}_j \leftarrow \left\{b_i \mid \mathbf{x}_{C(i)}^{(c)} \in \mathbb{M}_j\right\} \cup \left\{b_i \mid C(i) = -1, j = \min_{1\leq l\leq N_m} \|\mathbf{x}_i - \mathbf{m}_l\|\right\}$;

lower maximum cluster size to $P = 7$, accounting for approximately 12.5% of the clusters.

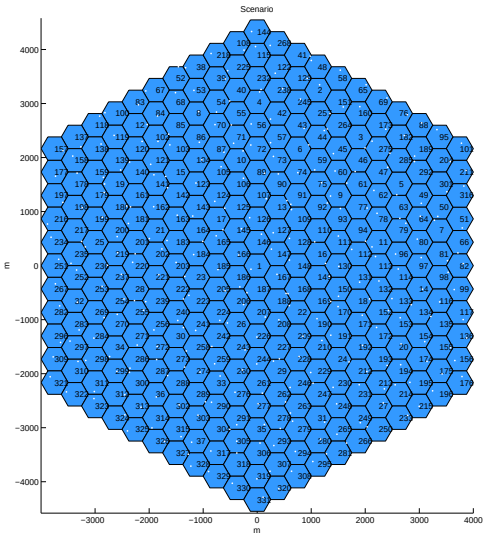


Figure 5.6: Example of a BTS deployment to illustrate algorithm 5.5. White dots represent user locations.

In figures 5.7 and 5.8, we can see the estimated PDF of the user locations as a contour plot and a 3D plot, respectively. It is interesting to observe how the estimated modes shown in figure 5.7, corresponding to local maxima in the kernel density estimate of figure 5.8, are located in regions where there are several users nearby.

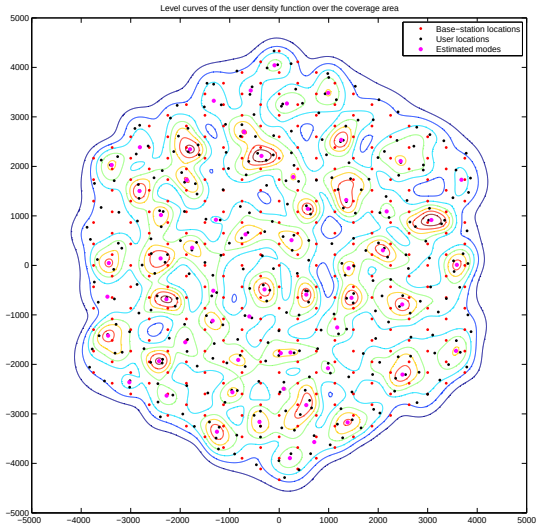


Figure 5.7: Contour plot of the kernel density estimate obtained for the scenario depicted in figure 5.6.

Figure 5.9 sheds light on the way algorithm 5.5 works. We can clearly see how each base-

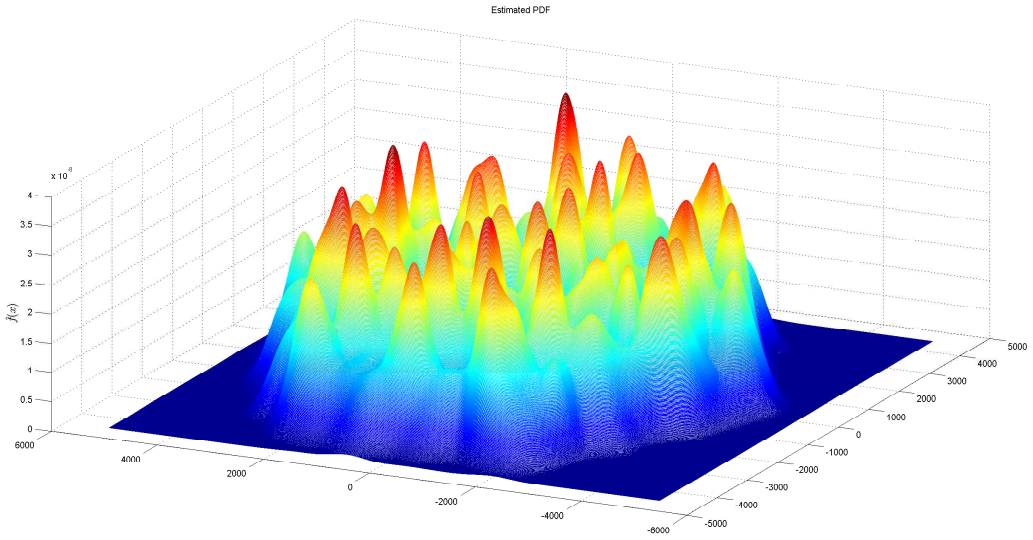


Figure 5.8: 3D plot of the kernel density estimate obtained for the scenario depicted in figure 5.6.

station "ascends" perpendicularly to the level curves of the kernel density estimate until reaching the nearest peak (mode). Also, it is very easy to see that, since several base-stations "meet" at each peak, the mean-shift procedure defines a natural partition of the set of base-stations.

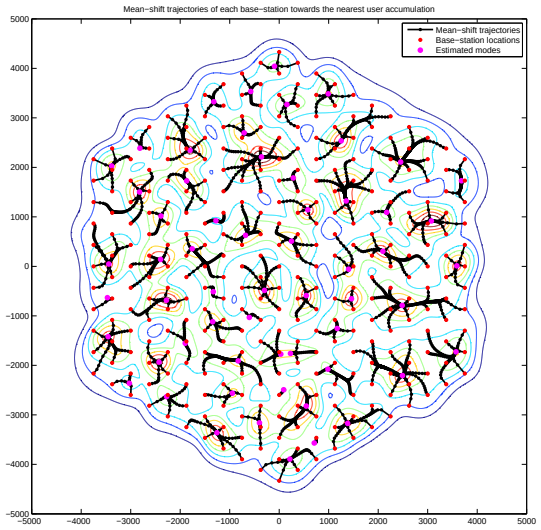


Figure 5.9: Example of a BTS deployment to illustrate algorithm 5.5. White dots represent user locations.

Finally, figure 5.5 shows the resulting clusters which were obtained by applying the algorithm. If the readers compare carefully figures 5.9 and 5.10, they can see how each attraction basin of figure 5.9 draws several base-stations, hence creating a cluster. However, we must point out

that, because the partition in figure 5.10 is the result of subsequently processing the partition obtained by the mean-shift base-station clustering algorithm according to the general structure shown in algorithm 5.1, there are some mismatches between the clusters defined by the modes and those shown here as the algorithm's output.

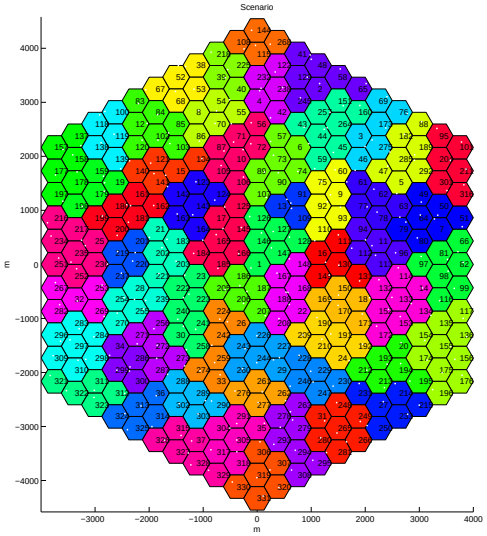


Figure 5.10: Resulting partition obtained by applying algorithm 5.5 for the scenario depicted in figure 5.6 with $L_{\max,a} = 7$, $L_{\max,b} = 10$ and $P = 7$. Under this settings in the generic structure 5.1, the resulting average and median cluster sizes of the depicted partition are smaller than those obtained using fixed hexagonal clusters with $T = 1$ tiers, but the system median rate was still improved by approximately 0.75 bits/s/Hz.

5.2.5.2 Parametrization

The first point to consider is that, as the reader has probably realized, mean-shift base-station clustering does not include any type of cluster size constraints in its formulation. This is because, unlike algorithms 5.3 and 5.4, when we designed this algorithm we were already using the generic structure in 5.1. In other words, mean-shift base-station clustering was specifically designed to be used as a "core clustering algorithm" within the aforementioned algorithmic structure, letting other subsequent steps take care of satisfying the constraints by refining the coarse partition provided by this algorithm.

One of the main parameters of all mean-shift based algorithms is the kernel profile $k(x)$ which generates the multivariate-kernel responsible for the kernel density estimation. In section 4.5 we studied in detail several typical choices and their corresponding strengths and weaknesses.

In our particular case, we have empirically checked that the exponential profile, giving rise to the usage of multivariate Gaussian kernels, vastly outperforms the rest of alternatives. Our intuition for the reason behind that is based on the following observation. We are not actually

dealing with a clustering problem *per-se*. Unlike any typical data analysis application of the mean-shift algorithm, we do not really want to estimate the underlying PDF of the user locations but, rather, we want to "misuse" the kernel density estimation to produce an instantaneous weighting of the coverage area which assigns big weights to points with many users nearby and small weights to points far away of most users. For any machine learning engineer, this is a rather strange situation in which one of its recurrent nightmares, overfitting, is actually beneficial. Actually, the situation is so ironic that, if we replaced the overfitted kernel density estimate by the real PDF of the user locations, which would be an ideal situation for machine learning problems, our algorithm would break down as the real PDF does not contain the "instantaneous" information we are interested in.

Nonetheless, restricting ourselves to using multivariate Gaussian kernels does not limit in any way the flexibility of our mean-shift base-station clustering algorithm. The set of bandwidth matrices $\{\mathbf{H}_i\}_{i=1}^{M}$ with $\mathbf{H}_i \in \mathbb{R}^{2\times 2}$ alone provides tons of degrees of freedom. Actually, in the most general case, we have too many parameters to be optimized. Because of this, we will concentrate in simpler solutions:

Single scalar bandwidth: All the cells employ the same radially symmetric kernel,

$$\mathbf{H}_i = h^2\mathbf{I}_2 \; \forall \, i = 1, \ldots, M \tag{5.10}$$

Per-cell scalar bandwidth: Each cell employs its own radially symmetric kernel,

$$\mathbf{H}_i = h_i^2\mathbf{I}_2 \; \forall \, i = 1, \ldots, M \tag{5.11}$$

Moreover, we have also explored an heuristic which attempts to provide an adaptive, semi-automatic parametrization tuning of the algorithm. The idea is to setup h_i^2 proportionally to the average squared distance between the i-th user and its K-nearest neighboring users, with a scalar proportionality factor α to be optimized globally. Mathematically:

$$h_i^2 = \frac{\alpha}{K} \sum_{j \in \mathbb{KNN}(i)} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \tag{5.12}$$

Where $\mathbb{KNN}(i)$ denotes the set of indexes of the K-nearest neighbors of the i-th user. Note that, as we discussed in section 4.5, this heuristic can also be applied in data-analysis yielding good results. Moreover, if Euclidean metric was not appropriate for a given data set, we can replace it by any other dissimilarity metric such as the Mahalanobis distance. However, since path loss is proportional to the Euclidean distance, this metric seems to be the best choice in our context.

Coming back to the idea that, in our particular case, overfitting is a very interesting situation, we reach the conclusion that the set of bandwidths $\{h_i\}_{i=1}^{M}$ is bound to have a tremendous impact on the performance of the algorithm, since it can drastically modify the "instantaneous weighting of the coverage area" we previously discussed, hence changing a lot the valleys and hills which

control to which locations the base-stations "fall". Because of this, we believed that properly tuning $\{h_i\}_{i=1}^{M}$ is so important, that it is interesting to create a complete genetic-algorithm just for that purpose. That motivates our next base-station clustering algorithm.

5.2.6 Genetic mean-shift clustering with applications to base-station coordination in MIMO wireless cellular systems

As we anticipated in the previous section, one of the contributions of this project has been to design a genetic mean-shift clustering algorithm in which we optimize the set of bandwidth matrices by using evolutionary computation. In order to keep a reasonable trade-off between flexibility and computational complexity, we will focus on radially symmetric kernels but still allowing each sample to have a different bandwidth. In other words, we are in the case we introduced as "per-cell scalar bandwidth parameters" in section 5.2.5.2.

Considering that the data set to be used for kernel density estimation contains M observations, our current objective is to design a genetic algorithm which optimizes the set of scalar bandwidth parameters $\{h_i\}_{i=1}^{M}$. In this way, we expect that the partition obtained when using algorithm 4.13 with those bandwidth parameters is the best possible.

As the reader may have already realized, in this section we are actually discussing a general purpose clustering algorithm, which could perfectly be applied for data analysis as an improvement over algorithm 4.13. Because of that, it can be thought that this algorithm belongs to chapter 4. Nonetheless, there are two fundamental reasons why we included it here.

On the one hand, even though it is indeed a general purpose clustering algorithm, its development was motivated within the context of base-station clustering for coordinated MIMO transmission. Besides, by including our genetic mean-shift clustering algorithm here, we keep all our contributions for clustering in chapter 5.

Sadly, as we discussed in chapter 4, the complexity of genetic algorithms is tremendous. As a consequence, genetic mean-shift clustering cannot be actually implemented in a real mobile communication system since it will never by able to comply with the real-time requirements of such systems given the current computational power available.

This algorithm and evolutionary base-station clustering, which we will discuss in the next section, are actually the ones which have allowed us to achieve our greater success in this project: to prove that it is really possible to significatively improve the performance of any next generation mobile communications system by carefully designing the groups of base-stations which are allowed to coordinate. This allows us to motivate new lines of research that have been systematically ignored until now. Even more importantly, by using these algorithms, we can build an enormous data set of (scenario/quasi-optimal partition) pairs which can be used to work out better and faster algorithms by reverse-engineering. This is a future line of research that we just opened, all thanks to our two evolutionary computation based clustering algorithms.

We will now discuss the main characteristics of the genetic mean-shift clustering from the point of view of evolutionary computation.

Genetic representation: In genetic mean-shift clustering, we want to optimize the set $\{h_i\}_{i=1}^{M}$ of scalar bandwidth parameters to be used in algorithm 4.13. Because the optimization is to be carried over M real variables, the best choice is to employ real-valued chromosomes of fixed length M. Then, an individual will be represented by a chromosome $\mathbf{h} \in \mathbb{R}^M$ in such a way that the i-th bandwidth matrix for algorithm 4.13 can be obtained as $\mathbf{H}_i = (\mathbf{h}^i)^2\mathbf{I}$.

Population initialization: Each individual is initialized pseudo-randomly. The basic idea is that every initial chromosome $\mathbf{h}[i]$ is obtained using the K-nearest neighbors heuristic described in section 5.2.5.2. However, both the number of neighbors K involved in the calculation and the scalar parameter α are chosen randomly within a predefined range in independent draws for each individual.

Fitness function: When the algorithm is to be used for data-analysis, any typical clustering cost function such as the silhouette function or even the Rand index (in case a reference partition is available) can be used.

More interestingly, we already said in the introduction of this chapter that one of the fundamental differences between clustering for machine learning and clustering for coordination of base-stations was that, in the latter, we actually have very clear, objective measures to be optimized. Even though there are many possibilities, the most relevant ones are:

- Sum-rate of the system (equivalent to the average rate of the system).
- Median-rate of the system
- BER of the system (to be minimized) or average number of bits between errors, BER^{-1}, (to be maximized).
- Sum-MSE of the system (to be minimized) or any non-decreasing function of the total MSE, (to be maximized).
- Median-MSE of the system (to be minimized) or any non-decreasing function of the median MSE, (to be maximized).

Nonetheless, we can use any other fitness function as long as it accurately represents the performance achieved by the candidate solution.

Obviously, to be able to evaluate any of the previous functions, it is necessary to obtain first the precoder $\mathbf{W_{tx}}$ and receive filter $\mathbf{W_{rx}}$ of each of the clusters defined by the partition to be evaluated, according to some prefixed MIMO signal processing technique like those discussed in chapter 3.

Another fundamental consideration when genetic mean-shift clustering is used to group base-stations is that, to comply with the generic structure defined in algorithm 5.1, the fitness evaluation routine must carry out steps 2 to 4 of algorithm 5.1 before anything

else. In other words, we will be employing algorithm 4.13 as the so called "core clustering algorithm" in step 1 of 5.1. Actually, if we are picky, since we are now discussing within the context of clustering base-stations, we should probably say that we use algorithm 5.5 instead of algorithm 4.13.

In any way, the important conclusion is that, when the objects to be clustered are base-stations, the genetic algorithm optimizing the whole clustering algorithm is nothing but the first and most important step of our generic base-station clustering algorithm 5.1. Also, in this case, we must include the maximum cluster size constraint thresholds $L_{\max,a}$, $L_{\max,b}$ and number of exceptions P within the fitness evaluation routine, as it will be responsible for carrying out the step which modifies the partition to make it compliant with the specs.

Selection: We use either roulette wheel selection or rank-selection, as defined in section 4.6.1.3. Because we want to keep the population size m constant throughout generations, elitism is applied whenever the mating pool size e is smaller than the population size, $e < m$, by copying the $m - e$ fittest individuals to the next generation without modifying them at all.

Genetic operators: Our genetic operators are as those described in section 4.6.1.4 for real-valued genetic representation. In this algorithm, we employ both crossover and mutation combined.

Crossover is implemented as one of the following well-known crossover operators for real-valued chromosomes: one-point crossover, two-point crossover, uniform crossover and arithmetic crossover. For every pair of individuals in the mating pool, we throw a biased coin with heads probability p_c. If we get tail, the parents are copied onto the next generation. Otherwise, we randomly pick one of the four implemented crossover operators and use it to obtain the offspring.

On the other hand, we have implemented a single mutation operator which acts over the offspring resulting from the crossing over. The mutation operator chosen is the one described in section 4.6.1.4 for real-valued chromosomes, i.e. we consider for each gene a small probability p_m of adding a small random number ϵ drawn from a Gaussian distribution.

5.2.6.1 Pseudo-code

Just as in algorithm 4.13, we use two data sets: $\mathbb{Y}$ will be used for kernel density estimation whereas $\mathbb{X}$ will be used as the set of seeds for the gradient ascent procedure. It is precisely the latter the data set which is to be clustered. When applying the algorithm to clustering of base-stations for coordination in MIMO cellular systems, the data sets are to be interpreted just as in mean-shift base-station clustering, that is, $\mathbb{Y}$ is the set of user locations and $\mathbb{X}$ the set of base-station locations.

First of all, the algorithm begins by initializing a set of individuals encoding a diverse variety of candidate solutions. After that, we simulate generation after generation until the maximum

Algorithm 5.6: Genetic mean-shift clustering algorithm

input : A data set $\mathbb{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\} \mid \mathbf{y}_i \in \mathbb{R}^d$ to be used for kernel density estimation

input : A data set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \mid \mathbf{x}_i \in \mathbb{R}^d$ to be used as seeds in the gradient ascent procedure

input : Kernel profile $k(x)$

input : Population size m and mating pool size e

input : Overall fitness evaluation function $F(\mathbb{S}[i])$

input : Crossover probability p_c and mutation probability p_m

input : Number of generations to be simulated, T

output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of $\mathbb{X}$

Create an initial population of individuals $\{\mathbf{h}[i]\}_{i=1}^{m}$ as described in 5.2.6 ;
it $\leftarrow 1$
while it $\leq T$ **do**
    for $i \leftarrow 1$ **to** m **do**
        Obtain a candidate partition $\mathbb{S}[i]$ by applying the generalized mean-shift algorithm 4.13 with the per-sample scalar bandwidth parameters encoded in individual $\mathbf{h}[i]$;
        Evaluate the fitness of the i-th individual as $F(\mathbb{S}[i])$;
    end
    Apply selection to fill a mating pool of size e with individuals from the population ;
    Shuffle the mating pool and divide it in $e/2$ pairs of individuals ;
    Apply crossover to each pair with probability p_c to obtain two new individuals. Otherwise, copy the original pair as if they were the children ;
    Apply mutation to each individual in the offspring with probability p_m ;
    Replace the current generation with the offspring ;
    it $\leftarrow$ it $+ 1$;
end
Output the partition $\mathbb{S}[i^*]$ originated from running generalized mean-shift algorithm 4.13 with the fittest individual of the last generation ;

number of iterations T has been elapsed. Because our objective is to obtain an optimal partition, the fitness function is defined over the set of possible partitions of data set $\mathbb{X}$. However, our chromosomes do not encode the clusters themselves but the parameters controlling the clustering algorithm. This implies that in order to evaluate the fitness of a candidate solution, we first have to run the generalized mean-shift clustering with the parameters encoded in the chromosome, and only then the fitness of the corresponding individual can be assessed.

Besides, as we discussed in 5.2.6, when we deal with base-stations the fitness evaluation routine becomes much more complex. Because we want to use the generic structure in algorithm 5.1, to obtain the partition which is to be evaluated by the fitness function, running generalized mean-shift clustering with the parameters encoded in the chromosome is no longer enough. Indeed, we also need to execute all the three subsequent, partition-refining steps in 5.1. If we did not do that, our algorithm would be useless since we would not be including the maximum cluster size constraints in the pseudo-code. Alternative solutions like not following the generic structure of algorithm 5.1 and fixing the problem by including a penalization term in the fitness function to mark unfeasible individuals as unfit could work but, as we discussed in section 4.6.1.2, that's a suboptimal approach.

Because of that, if we cluster base-stations, we need to add the cluster size constraints defined by 5.1, $L_{\max,a}$, $L_{\max,b}$ and P, into the fitness evaluation routine.

Once the fitness of all individuals has been evaluated, we can apply the selection operator to pick e individuals to fill the mating pool. As we use 2-parent crossover operators, e should be an even number smaller or equal than the population size, m. In our experiments, we have used mostly rank-selection, but any other strategy like roulette wheel selection or fitness scaling can be employed too.

When the mating pool is complete, the individuals are randomly grouped in pairs in preparation for crossing over. For each pair, crossover occurs with probability p_c. If it does, one of the four crossover operators is randomly chosen and applied to generate two children. If it does not, the parents are copied instead. After that, mutation is applied gene by gene to the chromosomes of the offspring resulting of crossover.

At this point, we have e individuals for the next generation. This algorithm will keep the population size m constant for all generations. Hence, the remaining $m - e$ individuals are obtained through elitism.

The process is iterated and, after T generations have been simulated, the partition obtained using the parameters encoded in the fittest individual of the most evolved generation is the algorithm's final output.

5.2.6.2 Example

As an example, let us revisit the scenario depicted in figure 5.6 by applying our brand-new genetic mean-shift base-station clustering algorithm.

In figure 5.11, we show the evolution of the fitness as the generations go by. Two different

fitness functions are considered: the system's average rate and median rate. For each generation, we show the fitness values of both the least fit individual and the fittest individual as well as the average fitness of the population. Also, as a reference, we include the fitness achieved by two other partitions.

On the one hand, we considered a fixed scheme based on cellular geometry with fixed hexagonal cluster of 7 base-stations like those of figure 5.2. In this way, we can see whether our algorithms provide any advantage with respect to the schemes that researches have been using up to now in their works regarding MIMO coordination for mobile communications.

On the other hand, since this algorithm attempts to enhance one of the algorithms we previously designed, basic mean-shift base-station clustering, we also include the results obtained with the partition depicted in figure 5.10, which was achieved precisely using that algorithm for the same scenario.

Clearly, the fixed scheme is vastly outperformed by our algorithms when median-rate is to be optimized. However, if we focus on the average rate instead, it seems that basic mean-shift base-station clustering does not provide a significant advantage with respect to the fixed assignation scheme based on cellular geometry. But the reader should recall that our algorithm was tuned precisely to optimize the median rate and not the average rate. If we repeated the plot by generating another partition with basic mean-shift base-station clustering tuned to optimize the average rate, we would see a big gap between our algorithm and the fixed scheme. More details on this matter will be discussed with exhaustive simulations in the next chapter.

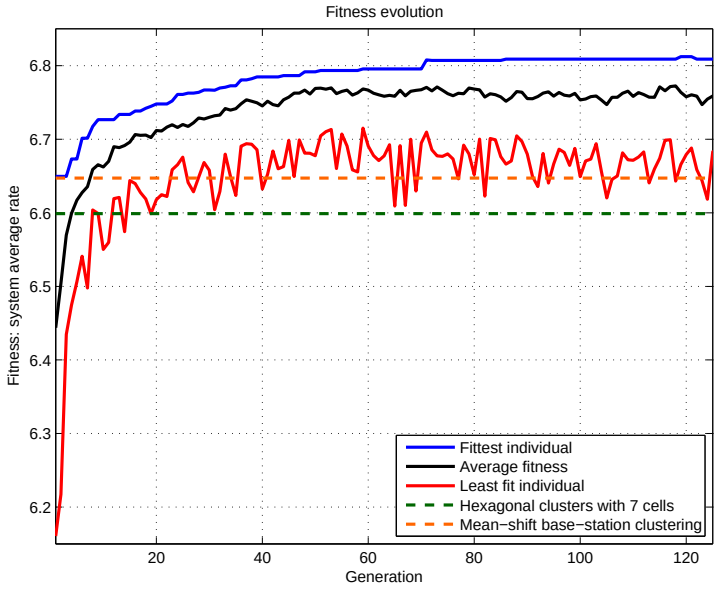
As far as this section is concerned, what we can observe is that for both choices of the objective measure to be optimized, the introduction of evolutionary computation provides a tremendous improvement. Regardless to say, the price to pay is that the algorithm's computational complexity becomes so big that introducing this algorithm in a real system is nowadays unfeasible. Nonetheless, as we previously said, the extreme performance achieved by this algorithm will allow researchers to use its results in order to design new, simpler algorithms.

Finally, the resulting partitions both for average-rate fitness and median-rate fitness are shown in figure 5.12.

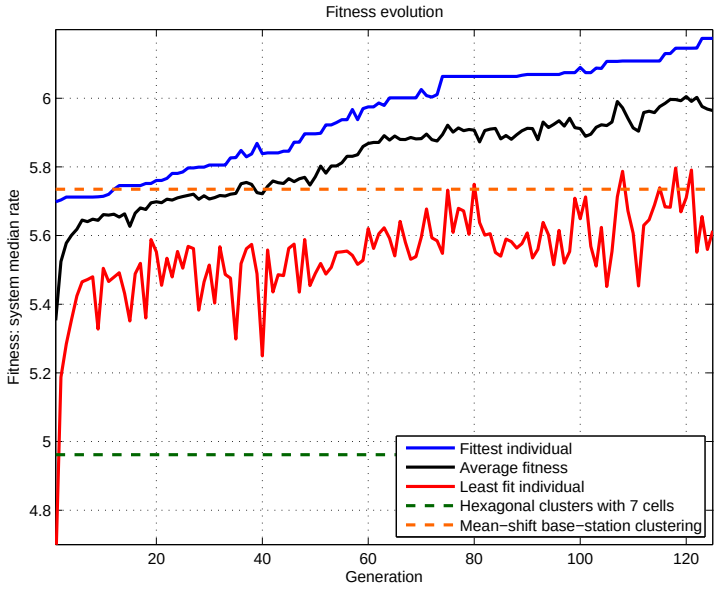
5.2.6.3 Parametrization

Genetic mean-shift clustering has a significatively simpler parametrization than basic mean-shift clustering. This should not be surprising since the whole idea behind this new algorithm is to make use of evolutionary computation to optimally tune the most important, and complicated to adjust, set of parameters in mean-shift based algorithms: the bandwidth matrices $\{\mathbf{H}_i\}_{i=1}^{N}$.

On the other hand, we still have to choose the kernel profile $k(x)$. All the discussions in that matter derived for the previous mean-shift based clustering algorithms the reader has already encountered in this project apply also to this algorithm. In data analysis applications, we have a very wide spectrum of possible choices and it is up to the engineer to discover which is the one which works better for the particular application they are interested in. On the other hand,

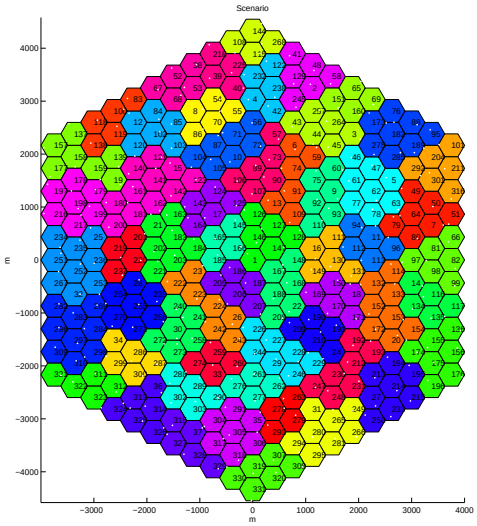


(a) Sum-rate fitness

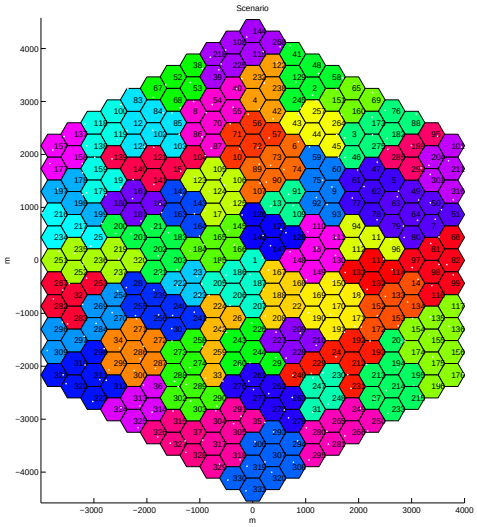


(b) Median-rate fitness

Figure 5.11: Evolution of the population's fitness with generations when applying genetic mean-shift base-station clustering for the scenario depicted in figure 5.6.



(a) Sum-rate fitness



(b) Median-rate fitness

Figure 5.12: Resulting partitions using genetic mean-shift base-station clustering for the scenario depicted in figure 5.6.

because of the same reasons we explained in section 5.2.5.2, when we want to cluster base-stations, Gaussian kernels outperform all the other choices we have studied in this document. Hence, in our final application, we use the exponential kernel profile to end up working with bivariate radially symmetric Gaussian kernels.

Last but not least, we have to realize that even though the introduction of a genetic algorithm to automatically tune the main parameters of mean-shift has succeeded in simplifying the parametrization of the mean-shift clustering routine, at the same time, we have also included a set of new parameters derived from the classic theory of genetic algorithms discussed in chapter 4. Again, as we discussed in section 4.6.1.5, little can be said a priori about parameters such as the population size m, the mating pool size e, the crossover probability p_c, the mutation probability p_m or the maximum number of generations T. Depending on the each application, the machine learning engineer shall adjust them mainly by a trial-and-error strategy. For further discussion of this topic, we refer the reader to the aforementioned section.

5.2.7 Evolutionary base-station clustering

When we started discussing on particular implementations of genetic algorithms for clustering, we stated that those could be roughly classified into two different families.

Some of them were actually designed as a way of enhancing another clustering algorithm by several means: optimizing its parameters, solving weakness such as sensitivity to random initializations or in any other way the designer of the algorithm is able to imagine. On the other hand, some others used evolutionary computation as the main driver of the solution. In this way even if some other clustering algorithms are used within the genetic clustering algorithm, they will have a secondary role like being yet another genetic operator. In other words, in the first family, the evolutionary part of the algorithm has a secondary role by aiding an already existing clustering algorithm which takes the main role. Opposed to that, in the second family, it is the evolutionary part of the algorithm the one taking the main role.

Clearly, our previous algorithm, genetic mean-shift clustering, belongs to the first family. The main role is carried out by the mean-shift clustering algorithm discussed in chapter 4, with the genetic part of the algorithm trying to optimize the set of bandwidth matrices.

In this section we develop a new algorithm, which we call evolutionary base-station clustering, following the path lead by genetic clustering algorithms belonging to the second family. This algorithm was not designed by modifying any already existing algorithm. It is the result of combining several ideas acquired by researching the different techniques proposed in the literature for tackling clustering problems with evolutionary computation and our own findings. However, if we were to point out some work to thank as the main source of inspiration, it would probably be the algorithm described in [48], FEAC. From that algorithm, we borrow several concepts we considered really interesting, such as the use of per-cluster fitness evaluations to create guided genetic operators or discarding the use of crossover operators hence using a mutation-only driven approach.

Evolutionary base-station clustering suffers from the same burden as genetic mean-shift clustering, i.e. its computational complexity makes its implementation in real mobile communication networks an unsurmountable task. However, as we discussed before, even if this algorithm may not be useful for mobile operators, they have proven to be fundamental for us researches working in the field of communications. The reason for that is that they hold an enormous amount of valuable information which has shed enough light on the matter of base-station coordination to open up new lines of research in our field.

We now describe the main characteristics of the algorithm.

Genetic representation: We employ label-based encoding as discussed in section 4.6.1.4 under the different integer-valued chromosomal representations.

As a recap, it is a very simple scheme in which each partition is encoded as the cluster assignation vector of the partition. Mathematically, an individual representing a partition $\mathbb{S} = \{\mathbb{S}_1, \ldots, \mathbb{S}_k\}$ of a set of M objects is encoded in a chromosome $\mathbf{c}$ of length M with integer-valued alleles within the alphabet $\{1, 2, \ldots, k\}$. In this way, $\mathbf{c}^i = j$ implies that the i-th object has been assigned to the j-th cluster in the partition represented by that particular individual.

Population initialization: As in genetic mean-shift clustering, each individual is initialized pseudo-randomly. However, here we use a different initialization routine since it has been tailored for the particular case of clustering base-stations.

Each individual is created independently from the others. First of all, the base-stations have their indexes shuffled randomly. Then, we iteratively pick the base-station not yet assigned to any cluster with the lowest index, and look for the closest $L_{\max,a} - 1$ base-stations not belonging to any cluster either in order to create a new cluster of $L_{\max,a}$ cells.

In our experiments, as a way to increase the diversity of the initial population to achieve a faster convergence, we also initialize some individuals using the previous algorithms we have designed with randomly chosen parameters. Of course, computationally-demanding algorithms like genetic mean-shift base-station clustering are not used for that purpose.

Fitness function: Any of the performance measures for distributed multi-user MIMO systems discussed in chapter 2 constitute perfectly valid candidates to use as fitness function. We particularly considered the same ones as in genetic mean-shift clustering:

- Sum-rate of the system (equivalent to the average rate of the system).
- Median-rate of the system
- BER of the system (to be minimized) or average number of bits between errors, BER^{-1}, (to be maximized).

- Sum-MSE of the system (to be minimized) or any non-decreasing function of the total MSE, (to be maximized).
- Median-MSE of the system (to be minimized) or any non-decreasing function of the median MSE, (to be maximized).

Changing the fitness evaluation routine can be done without any effort in our implementation, hence we don't need to limit ourselves to only those. However, whenever someone wants to try out a new fitness function, they must be aware that if it does not represent a proper measure of what they want the algorithm to achieve, the results may degrade noticeably.

As it was the case in genetic mean-shift base-station clustering, to evaluate a partition of a set of cells we need to compute the precoder $\mathbf{W_{tx}}$ and receiver filter $\mathbf{W_{rx}}$ cluster by cluster to build the system-wide precoding matrix and receive filter which allow obtaining the complete MIMO system model. Only then, we can evaluate any of the performance measures we may be interested in. Sadly, this fact is precisely the computational bottleneck of our two evolutionary clustering algorithms. Even if the computation of the system-wide matrices $\mathbf{W_{tx}}$ and $\mathbf{W_{rx}}$ is actually carried out in parallel as each cluster computes its own piece, the computational complexity is non-negligible. Then, if we realize that our evolutionary approaches require repeating that step for all individuals and in each generation, hence mT times, we start to understand why those algorithms are unfeasible for servicing real users in a real mobile communications network.

One of the main particularities of this algorithm is that we recover the concept of using also a per-cluster fitness evaluation, introduced in [48]. Thankfully, all the performance measures we have considered are actually defined at user level. Therefore, just as we compute the system-level measures by considering the particular values achieved by all users in the system, it seems natural to define the fitness of a cluster by doing the same but taking into account only the users belonging to that particular cluster.

As an example, if our objective is to maximize the median rate, the fitness of a given cluster is obtained as the median rate of all users belonging to that cluster and the overall fitness of the individual as the median rate of all users in the system.

Unlike the authors in [48], we do not require that the overall fitness is obtained as a sum of all per-cluster fitnesses weighted by the number of objects assigned to each cluster. However, the reader can readily check that, for some particular measures, like the average rate (sum rate) or the total MSE, our definition of per-cluster fitness also fulfills that property. On the other hand, fitness evaluation functions based on the median do not.

Similarly, we do not require any of the fitness values, overall or per-cluster, to be normalized within the interval $[0, 1]$.

Selection: In this algorithm we use rank-selection, defined in section 4.6.1.3. Also, the popula-

tion size m is kept constant throughout generations, and elitism can be optionally applied by setting a mating pool size $e < m$.

Genetic operators: Following the idea of FEAC, we discard the usage of crossover operators since they are quite difficult to design for chromosomes which directly encode partitions of a data set. Therefore, the mutation operator we defined has to carry out on its own the creation of the next generation of individuals as an evolved version of the current population.

Algorithm 5.7: Mutation operator for evolutionary base-station clustering

input : A set of base-stations $\mathbb{B} = \{b_1, b_2, \ldots, b_M\}$ and their corresponding locations $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \mid \mathbf{x}_i \in \mathbb{R}^2$

input : The partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ encoded by the individual to be mutated

input : The maximum distance between base-stations to consider that two cells are adjacent, $D_{\min}$

input : Lower and upper cluster size thresholds, $L_{\max,a}$ and $L_{\max,b}$, together with the number of clusters allowed to reach a size of $L_{\max,b}$ stations, P

input : Any additional information regarding the mobile communication scenario needed by the cluster splitting step

input : Overall fitness f of the individual to be mutated and per-cluster fitnesses $\{f_i^c\}_{i=1}^k$ of the clusters in the partition encoded by that individual

Compute the maximum number of clusters to be modified, $N_{\max}$, according to table 5.1 ;

Obtain the actual number of cluster to be modified, n, by drawing an integer from a uniform discrete distribution in the interval $[1, N_{\max}]$;

$n_{\text{mod}} \leftarrow 0$;

$\mathbb{I} \leftarrow \{1, 2, \ldots, k\}$;

while $n_{\text{mod}} < n$ **do**

  $i^* \leftarrow \min_{i \in \mathbb{I}} f_i^c$;

  Search for the set of neighboring clusters of cluster $\mathbb{S}_{i^*}$. Randomly pick some of them and store their indexes in collection $\mathbb{J}$;

  Randomly chose one of the cluster splitting algorithms described in section 5.4 and use it to split a cluster $\mathbb{S}_{i^*} \cup \bigcup_{j \in \mathbb{J}} \mathbb{S}_j$;

  Update the partition $\mathbb{S}$ by removing the clusters $\mathbb{S}_{i^*}$ and $\mathbb{S}_j \mid j \in \mathbb{J}$ and introducing the clusters which resulted from the previous splitting process ;

  $n_{\text{mod}} \leftarrow n_{\text{mod}} + |\mathbb{J}| + 1$;

  $\mathbb{I} \leftarrow \mathbb{I} \backslash \{\{i^*\} \cup \mathbb{J}\}$;

end

The mutation operator acts over a certain individual encoding a partition $\mathbb{S}$ of the set of base-stations with an arbitrary number of clusters k. Besides, we consider that the routine

implementing the mutation process has access to the fitness of the individual as well as to the fitness of each cluster $\mathbb{S}_i$ in the partition. Depending on which cluster splitting algorithms are considered in line 8, we may need additional inputs to carry them out apart from the parameters determining the cluster size constraints, $L_{\max,a}$, $L_{\max,b}$ and P. K-means cluster splitting does not require any additional information as the set of base-station locations $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \mid \mathbf{x}_i \in \mathbb{R}^2$ suffices. On the other hand, our cluster splitting algorithm based on hierarchical divisive spectral clustering does require as extra information a similarity matrix $\mathbf{W}$ computed with some of the similarity functions described in section 5.2.1. Finally, mean-shift cluster splitting needs, apart from the set of base-station locations $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \mid \mathbf{x}_i \in \mathbb{R}^2$, the corresponding set of user locations $\mathbb{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M\} \mid \mathbf{y}_i \in \mathbb{R}^2$.

The mutation operator begins by randomly selecting the number of clusters it will modify between within a range $[1, N_{\max}]$. In order to enhance the exploitation properties of the algorithm, the maximum number of clusters which can be modified, $N_{\max}$, is chosen as a percentage of the number of clusters in the partition depending on the fitness of the individual. In this way, the fittest individuals in the population will have a low $N_{\max}$, corresponding to a relatively small jump in the search space. On the other hand, the least fit individuals will have $N_{\max} = k$, so that the entire individual may actually change, hence potentially allowing a big jump in the search space. The exact way in which we obtain $N_{\max}$ as a function of the individual's fitness and the number of clusters in the partition is shown in table 5.1.

Rank in the population	$N_{\max}$
Top 10%	round$(0.2k)$
Between 10% **and** 30%	round$(0.4k)$
Between 30% **and** 60%	round$(0.6k)$
Between 60% **and** 80%	round$(0.8k)$
Bottom 20%	k

Table 5.1: Maximum number of clusters to be modified by the mutation operator depending on the rank of the individual within the population

It is important to consider that a high $N_{\max}$ does not imply that a big number of clusters will be modified. Similarly, even a relatively small $N_{\max}$ does not assure than very few clusters will be changed. This is because the number of clusters to be modified is chosen randomly in the range $[1, N_{\max}]$, being all numbers in the interval equiprobable. This property is fundamental, since it allows the algorithm to keep a healthy proportion of big and small jumps when exploring the search space. If we did never have big jumps in fit individuals, our algorithm would be prone to converge to local optimums. On the other hand, if we did not allow to take small jumps in unfit individuals, search spaces with very

steep hills and valleys would be difficult to tackle since, in those cases, the optimal point may be really close to a low-valued region.

Once we have the number of clusters that are to be modified stored in variable n, we initialize a counter of the number of clusters already modified, n_{mod}. Also, we create $\mathbb{I}$ to keep track of which clusters have not been modified yet.

At this point, we start an iterative procedure which is as follows.

First of all, the cluster not yet modified with the lowest fitness is chosen. We denoted its index as i^*. Then, we search for all the clusters which are neighbors of cluster i^*.

By neighbors, we understand those which are adjacent. A way to produce a mathematical definition out of that intuitive argument is to say that two clusters $\mathbb{S}_i$, $\mathbb{S}_j$ are neighbors when $\exists\, b_u \in \mathbb{S}_i, b_v \in \mathbb{S}_j \ : \ \|\mathbf{x}_u - \mathbf{x}_v\| < D_{\min}$ with $D_{\min}$ being the maximum distance for two cells to be considered as adjacent. In regular BTS deployments like the ones we consider, the distance between two adjacent cells is always twice the apothem of the hexagon representing the cell, that is, $D_{\min} = \sqrt{3}R_{\text{cell}}$.

To add more randomness to the mutation process, instead of keeping all the neighbors, some of them are discarded in a completely random way. The indexes of those which remain are temporally stored in variable $\mathbb{J}$.

The key step of our mutation algorithm comes next. We first obtain a mega-cluster agglutinating cluster $\mathbb{S}_{i^*}$ with its neighboring clusters annotated in $\mathbb{J}$ to subsequently split it with one of the algorithms of section 5.4. Both the actual algorithm and its parametrization are chosen randomly within some predefined ranges. The overall effect of this step is to redistribute all the cells belonging to the involved clusters in a different partition. This procedure allows to modify the partition locally but still keeping enough flexibility to explore the search space, thanks to including the neighboring clusters in the process. Note also that the number of clusters in the partition may change during this step, effectively increasing the genetic diversity within the population.

Finally, we update the count of modified clusters taking into account also the neighbors and remove all the clusters which have been modified from $\mathbb{I}$ to avoid changing them again.

This process is iterated until n or more clusters have had their cells redistributed.

As a technicality, the reader may have realized that it is possible that more than n clusters end up being modified, depending on the actual number of neighbors which are kept in the last iteration of the while loop. Even though it is irrelevant to the algorithm operation, we must accept that, strictly speaking, our previous definition of n is not fully correct. However, we prefer to keep it like that to avoid making the discussion more confusing.

Algorithm 5.8: Evolutionary base-station clustering algorithm

input : A set of base-stations $\mathbb{B} = \{b_1, b_2, \ldots, b_M\}$ and their corresponding locations $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \mid \mathbf{x}_i \in \mathbb{R}^2$

input : Population size m and mating pool size e

input : Fitness evaluation function which can obtain both the overall fitness of a partition, $F(\mathbb{S}[i])$, and the fitness of a given cluster in the partition, $F(\mathbb{S}_j[i])$. Both types of evaluation must be consistent which each other, that is, they have to quantify the same magnitude within the system

input : Number of generations to be simulated, T

input : Lower and upper cluster size thresholds, $L_{\max,a}$ and $L_{\max,b}$, together with the number of clusters allowed to reach a size of $L_{\max,b}$ stations, P

output: A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of the set of base-stations $\mathbb{B}$ satisfying the maximum cluster size constraints

Create an initial population of individuals $\{\mathbf{c}[i]\}_{i=1}^{m}$ as described in 5.2.7 ;
it $\leftarrow 1$
while it $\leq T$ **do**
    for $i \leftarrow 1$ **to** m **do**
        Take the partition $\mathbb{S}[i]$ encoded by the i-th individual and consider it as the output of step 1 within the generic structure of algorithm 5.1. Apply the subsequent partition-refining steps described in the aforementioned algorithm to obtain a refined partition which satisfies the maximum cluster size constraints $\mathbb{S}'[i]$;
        Update the i-th individual, that is, change the chromosome $\mathbf{c}[i]$ so that it now encodes partition $\mathbb{S}'[i]$;
        Evaluate the overall fitness of the i-th individual and the fitness of each cluster within the partition it represents ;
    end
    Apply selection to fill a mating pool of size e with individuals from the population ;
    Apply the mutation operator described as algorithm 5.7 to each individual in the mating pool ;
    Replace the current generation with the mutated individuals ;
    it $\leftarrow$ it $+ 1$;
end
Output the partition $\mathbb{S}[i^*]$ encoded by the fittest individual of the last generation ;

5.2.7.1 Pseudo-code

The algorithm's pseudo-code follows the skeleton of any generic genetic algorithm with minor changes.

It beings by initializing the population at random. After that, the iterative process in which generations are simulated one after the other begins.

Prior to fitness evaluation, we must refine the partitions to ensure that they satisfy the required size constraints. For that, we apply steps 2 to 4 in 5.1 to the partition encoded by each individual. This is equivalent to using the complete structure 5.1, with the mutation operator acting as the "core algorithm" for step 1.

Once all individuals have been updated to represent the refined, constraint compliant partitions, we evaluate their fitness and the fitness of the clusters in their respective partitions.

The mating pool is then filled with e individuals, chosen from the population according to rank-selection. After that, each individual in the mating pool is mutated using the operator described in algorithm 5.7.

Finally, the mutated individuals are introduced as the next generation, with the remaining $m - e$ individuals, if any, being chosen by elitism as the fittest individuals in the current population.

The process is iterated until T generations have been simulated and the partition encoded by the fittest individual of the most evolved generation is chosen as the algorithm's final output.

5.2.7.2 Example

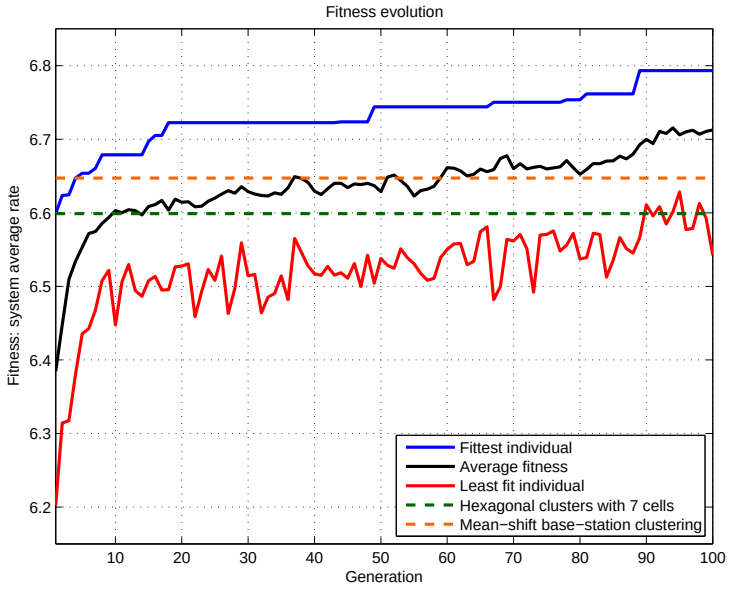
We will now continue the example discussed for the genetic mean-shift base-station clustering algorithm by applying evolutionary base-station clustering in the same scenario, that is, the one depicted in figure 5.6.

As before, figure 5.13 illustrates how the fitness evolves with the generations. We consider the same fitness functions than we did for genetic mean-shift clustering: the system's average rate and median rate. Again, the fitnesses included in the plot correspond to the least fit individual, the fittest individual and the population average. We also include the same references: the fixed scheme with hexagonal clusters of 7 BTS of figure 5.2 and the partition achieved by basic mean-shift clustering, tuned for median rate optimization only.

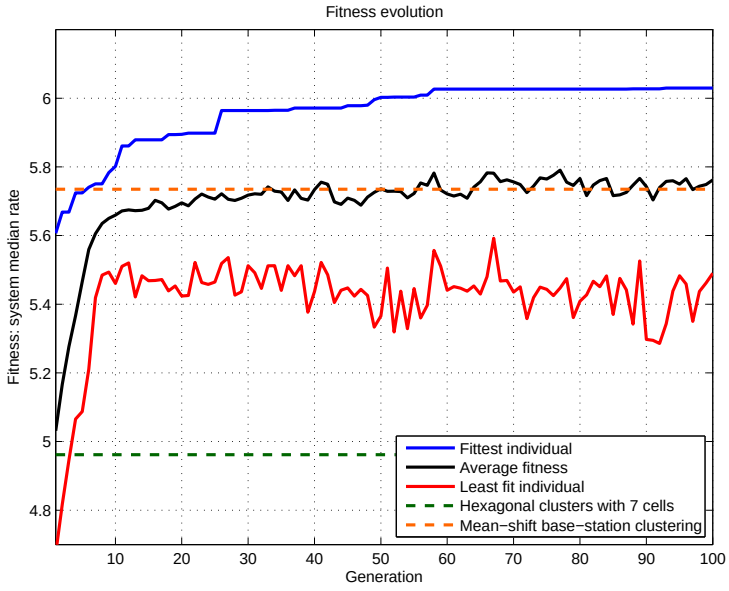
We can see that the behavior of this new algorithm is similar to those of genetic mean-shift base-station clustering, introducing an increase in the system's throughput. To end this example, we also include the resulting partitions in figure 5.14 for both choices of the fitness function.

5.2.7.3 Parametrization

The main parameters which can be tuned in this algorithm are those of genetic algorithms, which were discussed in section 4.6.1.5. Since the evolutionary part of the algorithm is actually

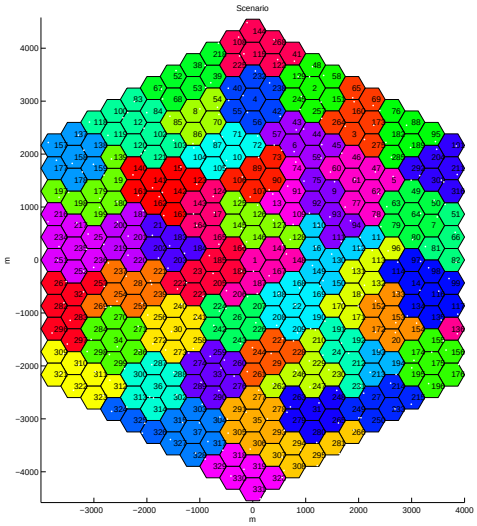


(a) Sum-rate fitness

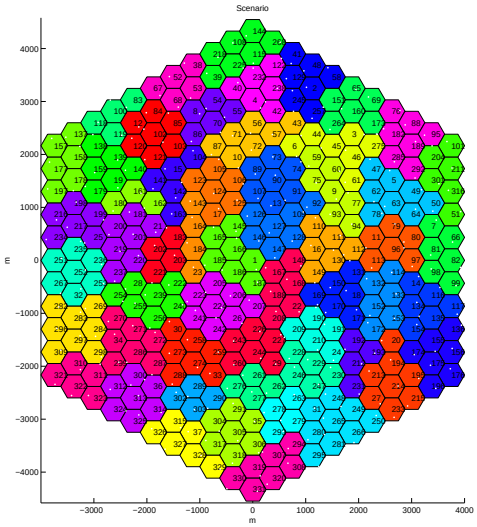


(b) Median-rate fitness

Figure 5.13: Evolution of the population's fitness with generations when applying evolutionary base-station clustering for the scenario depicted in figure 5.6.



(a) Sum-rate fitness



(b) Median-rate fitness

Figure 5.14: Resulting partitions using evolutionary base-station clustering for the scenario depicted in figure 5.6.

the one obtaining the partitions and the splitting cluster algorithms are randomly parametrized, there are little extra factors to be tuned.

5.3 Separation of connected components

During the second step of the generic base-station clustering algorithm 5.1, the partition is refined so that all its clusters are connected in a geographical sense.

To properly define what we understand by that, let us recall the concept of adjacent or neighboring cells we previously introduced.

When dealing with a regular base-station deployment like any of the ones we have depicted along this project, two cells are adjacent if the hexagons representing them "touch each other", that is, if they have a common side. It is very easy to check that for regular scenarios based on cellular geometry any given cell is adjacent to exactly six cells. Moreover, the distance between the centers of any two neighboring cells if always twice the apothem of the hexagon representing the cell, that is, $D_{\min} = \sqrt{3}R_{\text{cell}}$. However, in more realistic scenarios, the deployment of base-stations will still be a kind of grid, but the spacing between cells won't be so regular. Therefore, it will be necessary to define $D_{\min}$ according to the average distance between cells, adding also a certain tolerance to account for possible variations and irregularities.

In the end, we say that two cells are neighbors or that two cells are adjacent if their respective base stations are at most $D_{\min}$ meters apart. That is, cells i and j are neighbors if and only if $\|\mathbf{x}_i - \mathbf{x}_j\| \leq D_{\min}$. Once we have properly defined the concept of cell adjacency, we can provide a precise definition of connectedness in a geographical sense.

We say that a cluster $\mathbb{S}_a$ of base-stations in a mobile communications network is connected in a geographical sense if and only if for any two cells i and j belonging to $\mathbb{S}_a$ there exists a path joining them jumping between adjacent cells belonging to that same cluster. Mathematically, we can formulate that idea borrowing some ideas on graph theory from section 4.4.1. Indeed, we can use the base-station locations of cells within the cluster $\{\mathbf{x}_i\}_{i\in\mathbb{S}_a}$ to build an ϵ-neighborhood graph with $\epsilon = D_{\min}$ under the distance metric induced by the Euclidean norm. In that case, the connectedness concept for such a graph would be totally equivalent to our definition of connectedness in a geographical sense.

This motivates algorithm 5.9. As we can see, its pseudo-code is easy to follow.

The only inputs it requires are the partition $\mathbb{S}$ to be refined and, in order to check connectedness of the clusters in such partition, the base station locations $\mathbb{X}$ and the distance $D_{\min}$ which defines whether two cells are neighbors or not. The refined partition $\mathbb{S}'$ is obtained through an iterative process in which each cluster $\mathbb{S}_i$ in the original partition is inspected.

First of all, an ϵ-neighborhood graph is built using the base-station locations of cells belonging to the cluster under study, setting $\epsilon = D_{\min}$. We will represent the graph by its unweighted adjacency matrix $\mathbf{W}_i$.

The next and fundamental step is to use that adjacency matrix to divide the original cluster

Algorithm 5.9: Partition refining algorithm to ensure geographical connectedness for all clusters

input : A set of base-stations $\mathbb{B} = \{b_1, b_2, \ldots, b_M\}$ and their corresponding locations $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \mid \mathbf{x}_i \in \mathbb{R}^2$

input : A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of the set of base-stations $\mathbb{B}$

input : The maximum distance between base-stations to consider that two cells are adjacent, $D_{\min}$

output: A refined partition $\mathbb{S}' = \{\mathbb{S}'_1, \mathbb{S}'_2, \ldots, \mathbb{S}'_{k'}\}$ of the set of base-stations $\mathbb{B}$ containing only geographically connected clusters

$\mathbb{S}' \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** k **do**
  Obtain the associated unweighted adjacency matrix $\mathbf{W}_i$ associated to the ϵ-neighborhood graph built with points $\{\mathbf{x}_j\}_{j \in \mathbb{S}_i}$ and $\epsilon = D_{\min}$;
  Compute the connected components $\mathbb{C} = \{\mathbb{C}_1, \ldots, \mathbb{C}_{n_i}\} \; : \; \mathbb{S}_i = \bigcup_{j=1}^{n_i} \mathbb{C}_j, \sum_{\{u \in \mathbb{C}_a\}} \sum_{\{v \in \mathbb{C}_b\}} (\mathbf{W}_i)^{u,v} = 0$ of the graph represented by $\mathbf{W}_i$;
  $\mathbb{S}' \leftarrow \mathbb{S}' \cup \mathbb{C}$;
end

in its connected components. For that, we could use the properties of the graph Laplacian matrix discussed in chapter 4. On the one hand, we saw that the algebraic multiplicity of eigenvalue 0 was equal to the number of connected components in a graph. Moreover, the associated eigenvectors where the indicator vectors of each connected component. However, it is better to use an algorithm designed specifically for that purpose thus avoiding any spectral calculations, such as Tarjan's strongly connected components algorithm, [49].

In any way, no matter what algorithm we use for that purpose, once we have the connected components $\mathbb{C} = \{\mathbb{C}_1, \ldots, \mathbb{C}_{n_i}\}$ of the original cluster under study $\mathbb{S}_i$, they are added to the refined partition. Note that, if the original cluster was already geographically connected, $\mathbb{C} = \{\mathbb{S}_i\}$, the cluster would be copied unchanged from the input partition into the output partition. If not, the original cluster is split, with each geographically connected component being a different cluster in the refined partition.

Once all clusters in the input partition $\mathbb{S}$ have been processed, the algorithm ends and the refined partition $\mathbb{S}'$ is outputted.

5.4 Limiting cluster size: cluster splitting algorithms

Probably, the most important step of algorithm 5.1 after the initial sketch of the partition carried out during step 1 is the one we are going to discuss now. Unlike steps 2 and 4, which merely

attempt to refine the partition to slightly improve its performance, step 3 is fundamental as it is the one ensuring that the partition satisfies the maximum cluster size constraints.

Algorithm 5.10: Partition refining algorithm to split clusters violating the maximum cluster size constraint

input : A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of the set of base-stations $\mathbb{B}$, possibly violating the maximum cluster size constraints

input : Lower and upper cluster size thresholds, $L_{\max,a}$ and $L_{\max,b}$, together with the number of clusters allowed to reach a size of $L_{\max,b}$ stations, P

input : Any additional information regarding the mobile communications scenario needed by the cluster splitting step

output: A refined version of the original partition $\mathbb{S}$, $\mathbb{S}' = \left\{\mathbb{S}'_1, \mathbb{S}'_2, \ldots, \mathbb{S}'_{k'}\right\}$, such that $|\mathbb{S}'_i| \leq L_{\max,b}\ \forall i$ and $\sum_{i=1}^{k'} [|\mathbb{S}'_i| > L_{\max,a}] \leq P$, with $[\bullet]$ denoting the conditional operator which evaluates to 1 if its argument is true and 0 otherwise

1 Obtain the set of indexes $\mathbb{I}$ of all clusters $\mathbb{S}_i \in \mathbb{S}$ which violate the maximum cluster size constraints ;

2 **while** $\mathbb{I} \neq \emptyset$ **do**

3 Split a cluster $\mathbb{S}_i\ :\ i \in \mathbb{I}$ using one of the cluster splitting algorithms described in section 5.4.1 to obtain several sub-clusters which contain less than $L_{\max,a}$ cells each ;

4 Update the partition $\mathbb{S}$ to reflect the previous change ;

5 Reevaluate the set of indexes $\mathbb{I}$ of all clusters $\mathbb{S}_i \in \mathbb{S}$ which violate the maximum cluster size constraints ;

6 **end**

In order to implement such step, we propose the algorithm 5.10. It is an extremely simple iterative procedure in which one cluster violating the constraints is split in each iteration until the specifications are satisfied.

One of the most important decisions is the precise definition of the maximum cluster size constraints and the way of ordering the indexes of clusters violating such constraint within the collection $\mathbb{I}$.

In this project, as we previously discussed, we work with a flexible scheme. We consider a lower cluster size threshold, $L_{\max,a}$, which most clusters must satisfy. Nevertheless, in order to add some tolerance to compensate the existence of clusters with less than $L_{\max,a}$ cells in the partition, we also allow a small number P of clusters to have contain more than $L_{\max,a}$ cells. Precisely, at most as many cells as indicated by the upper cluster size threshold $L_{\max,b}$, which cannot be surpassed in any way.

However, extending algorithm 5.10 and, thus, all our base-station clustering algorithms, to work with other type of constraints is as straightforward as modifying the routine which searches within the partition $\mathbb{S}$ the set of clusters violating whatever constraints we have defined.

With our particular choice of constraints, there is a subtle point we have to clarify in that

sense. When obtaining the set of indexes $\mathbb{I}$ of all clusters $\mathbb{S}_i \in \mathbb{S}$ which violate the maximum cluster size constraints, there are two types of clusters contributing to $\mathbb{I}$. On the one hand, we have those which directly violate the upper cluster size threshold $L_{\max,b}$. Those clusters need to be definitely split, hence they will always be appended to $\mathbb{I}$. However we have an ambiguity whenever there are more than P clusters $\mathbb{S}_i$ in the partition $\mathbb{S}$ satisfying that $L_{\max,a} < |\mathbb{S}_i| \leq L_{\max,b}$. Precisely, if there were $P' > P$ clusters in such conditions, we would need to choose $P' - P$ clusters to be split. We consider two main choices which the user can decide.

On the one hand, splitting the biggest $P' - P$ clusters $\mathbb{S}_i$ satisfying $L_{\max,a} < |\mathbb{S}_i| \leq L_{\max,b}$ allows to achieve a more balanced partition in which most clusters have a similar size. This approach intuitively tends to favor the system median rate, sacrificing the sum-rate a bit.

On the contrary, if we split the smallest $P'-P$ clusters $\mathbb{S}_i$ satisfying $L_{\max,a} < |\mathbb{S}_i| \leq L_{\max,b}$, we obtain a more unbalanced partition which contains some big but constraint compliant clusters. This allows to have some clusters with a lot of interference reduction, hence favoring the system sum-rate but penalizing a bit the median-rate since we will also have a bigger number of small clusters.

In any way, no matter which criterion is chosen, once the set of clusters indexes to be split $\mathbb{I}$ is chosen, the algorithm simply picks one and splits it using one of the cluster splitting algorithms described in section 5.4.1. This is a fundamental step since it allows to modify the original cluster to obtain several sub-clusters which satisfy the maximum size specifications. Depending on the particular algorithm chosen, some additional inputs such as the set of user locations of a certain between cell similarity matrix $\mathbf{W}$ may be needed.

Note that in algorithm 5.10, only one cluster in the set $\mathbb{I}$ is split per iteration. After that, the partition is updated, $\mathbb{I}$ is reevaluated, and the process repeats itself until the maximum cluster size constraints are properly satisfied by the partition. One of the advantages of using this iterative scheme instead of directly iterating over the set $\mathbb{I}$ previously calculated is that we can allow a small probability of obtaining a sub-cluster with more than $L_{\max,a}$ cells as a result of the cluster splitting algorithms. If that happens, we know that cluster will be processed by the algorithm afterwards and the constraints will be eventually satisfied. Even though most of the cluster splitting algorithms we discuss in section 5.4.1 actually ensure that this situation is not going to arise, the one based on K-means has a relatively small probability of failing depending on the random initialization of the centroids. In this way, our iterative scheme makes that algorithm still usable allowing us to take advantage of its simplicity.

While discussing the genetic mutation operator we proposed for evolutionary base-station clustering, we introduced an interesting idea which we will recover now to motivate an alternative version of algorithm 5.10. We saw that when doing local modifications with the genetic mutation operator, it was interesting to increase a bit the scope, involving also the neighboring clusters to allow a more flexible redistribution of the cells. Now that we are splitting a certain cluster to satisfy the maximum cluster size constraints, we can also include that concept. The resulting algorithm is shown in 5.11.

Algorithm 5.11: Partition refining algorithm to split clusters violating the maximum cluster size constraint while redistributing cells with the immediate neighboring clusters

input : A set of base-stations $\mathbb{B} = \{b_1, b_2, \ldots, b_M\}$ and their corresponding locations $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \mid \mathbf{x}_i \in \mathbb{R}^2$

input : A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of the set of base-stations $\mathbb{B}$

input : The maximum distance between base-stations to consider that two cells are adjacent, $D_{\min}$

input : Lower and upper cluster size thresholds, $L_{\max,a}$ and $L_{\max,b}$, together with the number of clusters allowed to reach a size of $L_{\max,b}$ stations, P

input : Any additional information regarding the mobile communications scenario needed by the cluster splitting step

output: A refined version of the original partition $\mathbb{S}$, $\mathbb{S}' = \{\mathbb{S}'_1, \mathbb{S}'_2, \ldots, \mathbb{S}'_{k'}\}$, such that $|\mathbb{S}'_i| \leq L_{\max,b}\ \forall i$ and $\sum_{i=1}^{k'} [|\mathbb{S}'_i| > L_{\max,a}] \leq P$, with $[\bullet]$ denoting the conditional operator which evaluates to 1 if its arguments is true and 0 otherwise

1 Obtain the set of indexes $\mathbb{I}$ of all clusters $\mathbb{S}_i \in \mathbb{S}$ which violate the maximum cluster size constraints ;

2 **while** $\mathbb{I} \neq \emptyset$ **do**

3 Choose a cluster $\mathbb{S}_i\ :\ i \in \mathbb{I}$, and look for the set of its neighboring clusters storing their indexes in $\mathbb{J}$;

4 Split the mega-cluster formed by $\mathbb{S}_i$ and its neighbors, $\mathbb{S}_i \cup \bigcup_{j \in \mathbb{J}} \mathbb{S}_j$ using one of the cluster splitting algorithms described in section 5.4.1 to obtain several sub-clusters which contain less than $L_{\max,a}$ cells each ;

5 Update the partition $\mathbb{S}$ to reflect the previous change ;

6 Reevaluate the set of indexes $\mathbb{I}$ of all clusters $\mathbb{S}_i \in \mathbb{S}$ which violate the maximum cluster size constraints ;

7 **end**

5.4.1 Cluster splitting algorithms

Essentially, what we denote as "cluster splitting algorithms" are nothing but extensions of clustering algorithms being used with a slightly different purpose. If we think about it, the basic application of a clustering algorithm is to divide a set of objects into several groups. Then, if we had a cluster of cells containing too many cells hence making the coordination too expensive for our original specifications, a perfectly valid way to split it is by applying one of our base-station clustering algorithms taking into account only the cells belonging to the cluster to be divided.

However, several problems arise. On the one hand, with this approach we enter into a kind of vicious circle since we precisely said that we would develop most of our algorithms without explicitly including the maximum cluster size constraints as we expected to solve the problem in this step. Also, our algorithms worked well in a relatively big scenario with many cells. If only a

few base-stations belonging to a single cluster are available as the training set of the algorithm, their performance may degenerate a bit.

The second problem is the main motivating factor of the modification included in algorithm 5.11. By increasing the scope in which the cluster splitting algorithms operate, they operate over a bigger set of cells and perform their task slightly better, at the price of also a slightly increased complexity. However, according to our experiments, this problem is a minor issue. Whether we use algorithm 5.11 to deal with this problem, or simple ignore it and use the simplified version in algorithm 5.10 usually provides very similar results.

To solve the first problem, we propose several alternative solutions, which give rise to several cluster splitting algorithms.

K-means cluster splitting: One of them has already been discussed: by applying the cluster splitting algorithm several times, we radically enhance the algorithm's robustness.

This allows us to employ the simplest clustering algorithm possible, K-means, to divide the cluster in several pieces. To divide a certain cluster $\mathbb{S}_i$, we simply use the corresponding set of base-station locations $\{\mathbf{x}_j : b_j \in \mathbb{S}_i\}$ as the data set and choose K as $K = \text{round}\left(\frac{|\mathbb{S}_i|}{L_{\max,a}}\right)$. In this way, there is a big probability of obtaining clusters containing less than $L_{\max,a}$ cells. Nonetheless, depending on the random initialization of the centroids, we have absolutely no guarantees about that fact.

Thankfully, as the routine calling the cluster splitting algorithm processes the same clusters as many times as needed, eventually we will get a partition which satisfies the constraints. Strictly speaking, practical implementations of algorithms 5.10 or 5.11 will impose a limit in the maximum number of iterations to avoid getting stuck in an infinite loop in pathological cases. Hence, this approach has a vanishingly small probability of failure, even though according to our experiments such an event is extremely rare.

Hierarchical cluster splitting: Another alternative is to employ one of the two algorithms which actually included the maximum cluster size constraints in their formulation. Even if that was not their original purpose when we designed them, we ended up being lucky with the creation of this unexpected synergy between our algorithms.

To run this algorithm, we simply need to compute the similarity matrix $\mathbf{W}_i$ between the cells belonging to the cluster $\mathbb{S}_i$ to be split, according to any of the metrics discussed in section 5.2.1. With it, we run either hierarchical agglomerative base-station clustering or hierarchical divisive spectral base-station clustering with a maximum cluster size constraint $L_{\max,a}$ to obtain the sought result.

Mean-shift cluster splitting: Our final idea is based on a different concept: by carefully tuning the parameters of the clustering algorithm, we can try to achieve a partition of the original cluster with all its sub-clusters containing less than $L_{\max,a}$ cells.

For this purpose, we chose our basic mean-shift base-station clustering algorithm, 5.5. Under this approach, we compute per-sample scalar bandwidths according to the K-NN heuristic described in section 5.2.5.2. The value of K is fixed a priori, however, the global scalar parameter α is obtained through a bisection method to obtain the partition satisfying that all the resulting sub-clusters have less than $L_{\max,a}$ cells with the biggest median cluster size. In this way, the resulting division of the cluster not only is specification compliant but also achieves a very good performance. Possible modifications, such as trying to obtain the biggest average cluster size or other criteria can be readily included too.

5.5 Merging small neighboring clusters

In the previous step, we used some splitting cluster algorithms in order to divide clusters violating the size constraints. Even though we carefully designed them to try to produce meaningful and balanced sub-clusters, there are usually some resulting clusters which contain few cells. Moreover, because we processed each cluster independently, it may happen that two sub-clusters obtained by splitting two different clusters in the original partition end up being neighbors with the potential to be merged. Precisely, this will happen whenever they contain sufficiently few cells so that the cluster resulting from their potential union would still satisfy the maximum cluster size constraints. Moreover, because we want to keep the geographical connectedness obtained during the refining step carried out by algorithm 5.9, we strictly require the sub-clusters to be neighbors. Otherwise, they won't be merged even if they are small enough.

Even though in a regular execution of the clustering procedure there are few occurrences of this situation, as bigger clusters allow for bigger interference cancellation, it is beneficial trying to get as close as possible to the maximum cluster size specifications without exceeding them. Hence, we introduced this forth and final step of the generic structure used by all our base-station clustering algorithms shown in 5.1. Its aim is to carry out the process the previously motivated: to refine the partition by merging small neighboring clusters whenever joining them does not violate the maximum cluster size constraints.

In algorithm 5.12 we illustrate the pseudo-code to fulfill that task.

One important issue which we did not mention previously is that, potentially, we may have several, mutually exclusive ways to carry out the merging process.

As a simple example, imagine a situation in which the original partition contains 3 neighboring clusters, say $\mathbb{S}_a$, $\mathbb{S}_b$ and $\mathbb{S}_c$. Let their respective sizes be $|\mathbb{S}_a|$, $|\mathbb{S}_b|$ and $|\mathbb{S}_c|$ base-stations. Moreover, let us assume that they satisfy $|\mathbb{S}_a| + |\mathbb{S}_b| \leq L_{\max,a}$, $|\mathbb{S}_a| + |\mathbb{S}_c| \leq L_{\max,a}$ and $|\mathbb{S}_b| + |\mathbb{S}_c| \leq L_{\max,a}$ but $|\mathbb{S}_a| + |\mathbb{S}_b| + |\mathbb{S}_c| > L_{\max,a}$. In other words, we can merge any two of them, but not the three. Therefore, we should make a choice out of three different possibilities: merging $\mathbb{S}_a$ with $\mathbb{S}_b$ leaving $\mathbb{S}_c$ alone, merging $\mathbb{S}_a$ with $\mathbb{S}_c$ leaving $\mathbb{S}_b$ alone or merging $\mathbb{S}_b$ with $\mathbb{S}_c$ leaving $\mathbb{S}_a$ alone. Each possibility will lead to a different partition... which will achieve a slightly different performance.

Algorithm 5.12: Partition refining algorithm to merge neighboring clusters in the partition whenever doing so

input : A set of base-stations $\mathbb{B} = \{b_1, b_2, \ldots, b_M\}$ and their corresponding locations $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \mid \mathbf{x}_i \in \mathbb{R}^2$

input : A partition $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_k\}$ of the set of base-stations $\mathbb{B}$

input : The maximum distance between base-stations to consider that two cells are adjacent, $D_{\min}$

input : Lower maximum cluster size threshold $L_{\max,a}$

output: A refined partition $\mathbb{S}' = \{\mathbb{S}'_1, \mathbb{S}'_2, \ldots, \mathbb{S}'_{k'}\}$ of the set of base-stations $\mathbb{B}$ obtained by merging small neighboring clusters of the input partition $\mathbb{S}$ whenever doing so does not violate the lower maximum cluster size constraint

$\mathbb{S}' \leftarrow \mathbb{S}$;
do
  $\mathbb{S} \leftarrow \mathbb{S}'$;
  $\mathbb{S}' \leftarrow \emptyset$;
  $k \leftarrow |\mathbb{S}|$;
  $\mathbb{I} \leftarrow \{1, 2, \ldots, k\}$;
  for $i \leftarrow 1$ **to** k **do**
    if $i \in \mathbb{I}$ **then**
      if $|\mathbb{S}_i| < L_{\max,a}$ **then**
        Store in $\mathbb{J}$ the set of neighbors of cluster $\mathbb{S}_i$, $\{\mathbb{S}_j\}_{j=1}^{N_i}$, which satisfy $j \in \mathbb{I}$ and $|\mathbb{S}_i| + |\mathbb{S}_j| \leq L_{\max,a}$. ;
        if $\mathbb{J} \neq \emptyset$ **then**
          $j^* \leftarrow \min_{j \in \mathbb{J}} |\mathbb{S}_j|$;
          $\mathbb{S}' \leftarrow \mathbb{S}' \cup \{\mathbb{S}_i \cup \mathbb{S}_{j^*}\}$;
          $\mathbb{I} \leftarrow \mathbb{I} \backslash \{i, j^*\}$;
          continue;
        end
      end
      $\mathbb{S}' \leftarrow \mathbb{S}' \cup \mathbb{S}_i$;
      $\mathbb{I} \leftarrow \mathbb{I} \backslash \{i\}$;
    end
  end
while $\mathbb{S} \neq \mathbb{S}'$;

In conclusion, the procedure we proposed to carry out in this step is actually an optimization problem itself, potentially as complex as the whole base-station clustering problem.

However, we do not intend by any means to achieve the optimal solution. On the other hand, we are content with employing a more modest greedy search approach very easy to implement. Even if it won't achieve the optimal solution, it will still enhance the system's performance by exploiting better the maximum cluster size specifications, hence it is a useful step.

The algorithm we propose is much simpler than it seems at first sight. It is essentially an iterative approach in which merging sweeps are applied to the partition successively until it is no longer possible to join any two neighboring clusters without violating the size constraints.

In each sweep, we process the clusters in the current partition, $\mathbb{S}$, one by one within the for loop. For each cluster $\mathbb{S}_i$, we try to find if there exist any neighboring clusters within $\mathbb{S}$ small enough to be merged with the cluster currently under study. If there are, we pick the one which contains less cells and merge it with $\mathbb{S}_i$. We impose the restriction that any given cluster can only be merged once. We keep track of which clusters have already been joined with some other cluster in variable $\mathbb{I}$. Once all clusters have been processed, the current partition $\mathbb{S}$ is substituted by the modified partition $\mathbb{S}'$.

Note that, within the for loop, we process only the clusters in the current partition $\mathbb{S}$ and look for neighbors only within that partition. In other words, in each sweep, we only update the partition at the very end. This does not allow things such as merging a certain cluster $\mathbb{S}_i$ with a cluster which resulted from merging other two cluster $\mathbb{S}_j$, $\mathbb{S}_l$ during the same sweep. This is precisely the reason why we have to do several sweeps until convergence. Otherwise, we would be ignoring useful cases such as merging more than two neighboring sufficiently small clusters all together. The main motivation for implementing the algorithm like this is to avoid having to design a complicated procedure, in which the for loop iterated over a set of clusters which were changing as they are being processed.

Therefore, in our algorithm, as the sweeps go by, small neighboring clusters are successively merged until we eventually reach a state in which we cannot find any pair of adjacent clusters small enough to be joined. In that sweep, the resulting partition $\mathbb{S}'$ will be identical to the original partition $\mathbb{S}$ and the algorithm will halt.

By having studied this last step of the generic skeleton of our base-station clustering algorithms, we have completed the theoretical discussion on the proposed approaches for grouping base-stations in mobile communication systems employing coordinated MIMO transmission.

Chapter 6

Simulation results

During this chapter, we will design and execute a battery of exhaustive simulations to test the validity of the base-station clustering algorithms proposed during our research. These experiments aim at giving some insight about the usefulness of researching about optimal ways of grouping base-stations into clusters for MIMO coordinated transmission. This way, during the rest of this chapter, we will devote to discussing the performance of the base-station clustering algorithms introduced in chapter 5. We will begin by explaining in detail the particularities of the simulation scenario we have chosen, so that our results can be replicated in case it is desired. After that, we will show and discuss the results obtained from our tests and the relation we found between the achieved performance and the parametrization of each algorithm.

6.1 Simulation scenario

In this project, we will work will BTS deployments like the one illustrated in figure 6.1.

Unlike the case of GSM, representing cells with hexagons is merely a visual artifact which has no implications at all: all our clustering algorithms are based in one way or another on signal propagation, which we consider to be isotropic. Hence, cells are actually considered to be circular and overlapping. Nonetheless, as we discussed during chapter 5, extensions of our work to account for the usage of directive antennas is straightforward, implying a mere change in the parametrization of the algorithms.

More arguable is the choice of such a regular distribution for the base-station locations on the coverage area. While it is true that the layout of any real deployment of base-stations would certainly be more irregular, due to the constraints imposed by the existence of regulatory laws and structures such as buildings, we believe that a regular deployment based on cellular geometry provides a very reasonable approximation to any real distribution. This has been followed as a theoretical approach by a large number of books and research articles since the standardization of GSM and we will continue to do so. Moreover, any other approach would require particularizing all our work to the specific geography and urban topology of a certain city or region, whereas our aim here is to develop general-purpose algorithms.

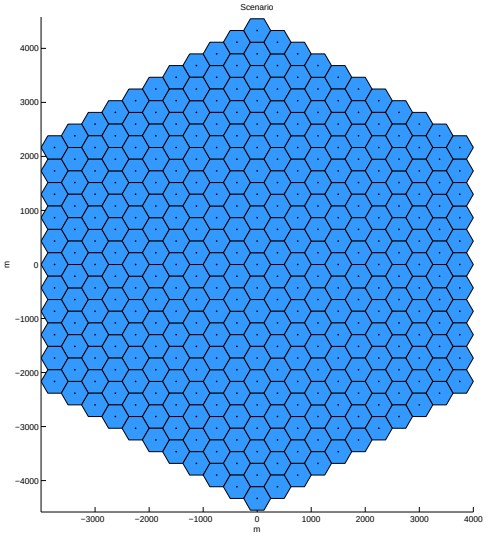


Figure 6.1: Approximation of a typical BTS deployment in cellular communications

Furthermore, by simulating a large number of stations in this tier-based structure, we can significatively reduce the border effects in the simulation. Those are simply due to the fact that base-stations in the outermost tiers suffer less interference than stations in the inner tiers. However, because the number of BTSs in a scenario with T tiers is $1+6\sum_{n=1}^{T} n = 3T^2+3T+1$ and the number of stations in the outermost tier is $6T$, the percentage of border stations decays asymptotically with the number of tiers as $2/T$. Besides, it is always possible to simply exclude those stations from the performance evaluation to avoid biasing the results. On the other hand, we must also consider that in a real scenario, due to the irregularities in the deployment we previously discussed, there will actually exist some "gaps", i.e. regions where base-stations are far enough to suffer only small interference levels. Because of that, if we discarded the outermost tier in the evaluation of the performance, our fully regular placement scheme can be regarded as a kind of worst-case scenario.

The results discussed in this chapter were obtained for the scenario depicted in figure 6.1. It was built by generating $T = 10$ tiers, hence it contains $M = 331$ cells. The set of base-station locations $\{\mathbf{x}_i\}_{i=1}^{M}$ are computed by regularly sampling an hexagonal grid with an step equal to the inter-site distance D_{int}. Mathematically, the base-station deployment process can be represented in matrix form as:

$$\mathbf{x} = D_{\text{int}} \begin{pmatrix} 0 & \frac{\sqrt{3}}{2} \\ -1 & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} \tag{6.1}$$

Where i and j are two integers representing the point in the grid being sampled. Depending on the range of values we consider for i and j, the shape of the resulting scenario changes. In our case, we chose them so that our base-station deployment consists of T tiers. However, it is

perfectly possible use a different shape, like a square or a rectangle, by changing the sampling limits.

It is straightforward to see that D_{int} is twice the apothem of the hexagon representing the cell. Hence, if the hexagons are depicted with a radius equal to the cell radius R_{cell}, then $D_{\text{int}} = 2R_{\text{cell}}$. Our simulations try to use typical parameters for LTE systems. In this case, assuming a standard LTE cell in a urban environment, we set $R_{\text{cell}} = 250m$. The resulting system covers then an area of approximately 31 km^2, representative of a typical medium-size town like Leganés.

Signal propagation is also simulated according to the 3GPP models for LTE. Distance dependent path loss is obtained with the following function of the distance d in meters:

$$\text{PL}(d) = 98.1 + 37.6 \log_{10}(d) \text{ dB} \tag{6.2}$$

Besides, we include the Rayleigh fading between the k-th receive antenna of the i-th user and the l-th transmit antenna of the j-th base-station by drawing a coefficient r_{ij}^{kl} from a circularly symmetric complex Gaussian distribution with unit power. Each element of the channel matrix $\mathbf{H}$ is then evaluated as:

$$(\mathbf{H})_{i,j}^{k,l} = \sqrt{\text{PL}\left(\|\mathbf{y}_i - \mathbf{x}_j\|\right)} r_{ij}^{kl} \tag{6.3}$$

Where $\mathbf{x}_j$ is the location of the j-th base-station and $\mathbf{y}_i$ the location of the i-th user, both with the coordinates expressed in meters. As we can see, our simulation is focused on the downlink channel. However, it would be possible to reproduce the simulations we carried out for the uplink, changing from a signal processing scheme based on precoding to one based on filtering, as discussed in chapter 3. Besides, signal propagation is considered to be purely isotropic. Therefore, the scenario in which we are testing our algorithms is more restrictive in terms of interference than scenarios with three sectors per cell, where the antennas would be assumed to transmit only with a beam-width of 120º hence significatively reducing the number of cells which interfere in a given location.

The characteristics of the base-stations and the user equipments are also chosen according to typical LTE specifications. Base-stations are considered to transmit their signal using antennas with a gain of 14 dBi and a maximum available power of 46 dBm. On the other hand, UE antennas are assumed to have neither gain nor losses, that is, they have a gain of 0 dBi. Besides, a 2×2 MIMO configuration was chosen, that is, each base-station has $t = 2$ transmit antennas and each user equipment has $r = 2$ receive antennas.

Another fundamental parameter when simulating a communications system is the noise level, N_0 or, more precisely, the signal-to-noise ratio, SNR, as it is the one which in the end determines the BER and the system's rate. However, as we discussed in chapter 2, the radio access network of any mobile communications system is usually limited by interference. In other words, the role of the SNR in point-to-point communication systems is substituted by the SNIR. Even more importantly, the assumption of universal frequency reuse inherent to 4G or 4.5G systems leads

to a situation where the noise power $P_{\mathrm{n}} = \mathrm{N}_0\mathrm{BW}$ is actually negligible in the evaluation of the SNIR, since the total interference can be approximately one or even two orders of magnitude greater.

Because of this, modifying the noise level N_0 in our simulations introduces relatively small changes in the system's performance, as long as it is not set to dramatically high levels by which the noise power artificially surpasses the total interference. Even more importantly, unlike in most point-to-point communication systems, there exists a saturation effect by which $\mathrm{N}_0 \to 0$ does not imply that the rate of the system grows without bound nor that the total MSE approaches 0. This is because, even if the noise power is set to 0, we still have the all the interference power keeping the SNIR at relatively modest levels.

By a similar reasoning, increasing the maximum available power for transmission in the base-stations does not lead to a big enhancement in the system's performance. While it is true that it rises the received signal strength, it also increases the interference by the same amount. Mathematically, multiplying the transmit power by a factor α modifies the SNIR as:

$$\mathsf{SNIR}' = \frac{\alpha P_{\mathrm{s}}}{P_{\mathrm{n}} + \alpha P_{\mathrm{int}}} = \frac{P_{\mathrm{s}}}{\frac{P_{\mathrm{n}}}{\alpha} + P_{\mathrm{int}}} \tag{6.4}$$

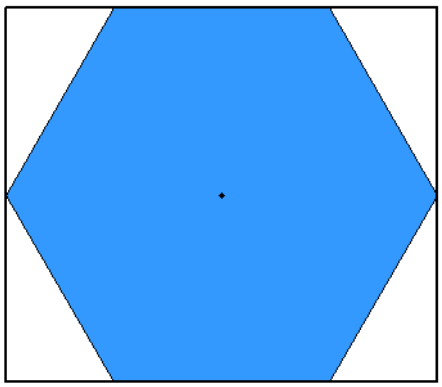
In other words, increasing the transmit power by $10\log_{10}(\alpha)$ dB simply makes the SNR that many dBs higher, but not the SNIR. Therefore, for scenarios in which the SNIR is mostly determined by the received interference, increasing the transmission power is not a big advantage.

Nonetheless, to be rigorous, we did introduce noise in our simulations. The noise vector $\mathbf{n}$ in the input of the UE antennas is assumed to be white and drawn from a complex Gaussian radially symmetric distribution. Its autocorrelation matrix is then of the form $\mathbf{R_n} = \sigma_n^2\mathbf{I}$ where σ_n^2 is obtained so that the signal-to-noise ratio at the cell border, $\rho = \frac{P_{\max}\mathrm{PL}(R_{\mathrm{cell}})}{\sigma_n^2}$, is 15 dB.

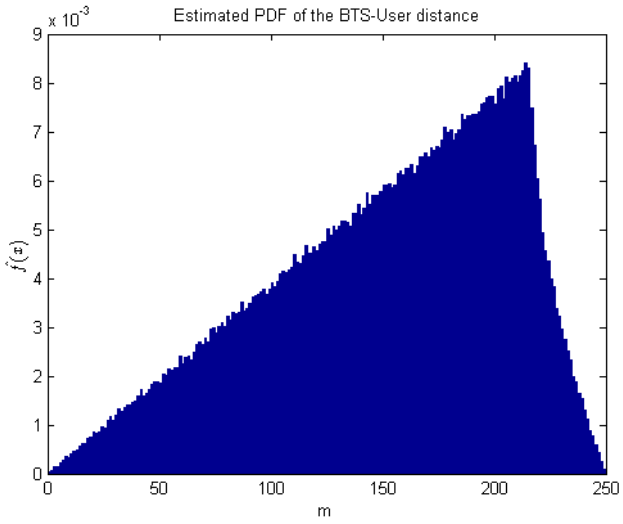
User placement is one of the most important issues in our simulation. As we previously discussed, our work assumes a transmission scheme based on OFDM, allowing multiple access of several users within a cell by statistical assignation of OFDM carriers, i.e. a form of Frequency Division Multiple Access (FDMA). In this chapter, we will simulate the complete system operating at a particular OFDM carrier. Hence, each cell will only serve a single user, as the others will have their transmission allocated at other non-interfering OFDM carriers. Strictly speaking, we should have considered some marginal probability of a certain cell having no users at all. However, we preferred to neglect that event and work with a worst-case scenario in which there are no empty cells in the system.

The particular location of the user being served within each cell is drawn from a uniform distribution of rectangular support with base $2R_{\mathrm{cell}}$ and height $\sqrt{3}R_{\mathrm{cell}}$, that is, the smallest rectangle containing the whole hexagon representing the cell. In figure 6.2(a), we show a graphical representation of the support of the underlying PDF from which the user location within a cell is drawn. If we happen to get an user outside the hexagon, we discard such location and another sample is generated until it is eventually within the hexagon area.

In figure 6.2(b), we show something quite interesting. As we can see, the user locations are



(a) Support of the underlying PDF of the user location within a cell. Probability density is constant within the depicted area. If the user location obtained is outside the hexagon, another location is drawn again.



(b) Histogram based PDF estimate of the distance $\|\mathbf{y}_i - \mathbf{x}_i\|$ between each UE and the base-station which is serving it.

Figure 6.2: Characteristics of the PDF for the user locations within a cell.

obtained in such a way that the PDF followed by the distance between any given base-station and the user which it is serving within its cell, $d = \|\mathbf{y}_i - \mathbf{x}_i\|$, is approximately a ramp until the distance equals the hexagon apothem, $d = \frac{\sqrt{3}}{2} R_{\text{cell}}$. Afterwards, it decays exponentially until $d = R_{\text{cell}}$, when the probability converges to zero.

An intuitive argument to explain why such probability density function appears to be a ramp

for most of its support can be developed as follows. Computing the probability of drawing a user location whose distance to the center of the cell belongs to the interval $[r, r + \Delta r] : 0 < r < \frac{\sqrt{3}}{2} R_{\text{cell}}$ is equivalent to computing the integral of the underlying PDF of the user locations within a ring of radius r and width Δr. Since the area of such a ring is $\pi\left((r + \Delta r)^2 - r^2\right) = 2\pi r\Delta r + \pi\Delta r^2$ and the probability density within the square depicted in figure 6.2(a) is constant, let's say $f(\mathbf{x}) = c$, then the probability we were looking for is $2\pi cr\Delta r + \pi\Delta r^2 c$. Dividing by Δr and letting $\Delta r \to 0$ to obtain the density of probability we can see that the PDF of the distance between the user location and the cell center for $0 < r < \frac{\sqrt{3}}{2} R_{\text{cell}}$ is $2\pi cr$, which matches the empirical results. A more rigorous way of obtaining that result could be developed by computing the corresponding integral using polar coordinates. In that case, it would be the Jacobian associated to the change of variables the responsible for introducing the linear dependence of the PDF with r.

Similarly, the exponential decay is caused by the particular shape of the hexagon. Because of the way this polygon distributes its area, the set of points which fulfill that the distance between them and the center of the hexagon is greater than the apothem but smaller than the radius defines a surface which is quite small. Therefore, as user locations are initially drawn for a uniform distribution, it is unlikely that we get users in that regions many times.

Finally, it is obvious that the maximum possible distance is $d = R_{\text{cell}}$, hence the probability for $d > R_{\text{cell}}$ has to be zero.

Nonetheless, the most important consideration regarding this PDF is the fact that there is a much higher probability of getting users quite far away from the base-station than nearby. As a consequence, we can say that our simulations are usually modeling unfavorable scenarios.

The last key choice to be discussed is the particular MIMO signal processing scheme employed to obtain the linear precoding matrices $\mathbf{W_{tx}}$ and receive filters $\mathbf{W_{rx}}$ for each cluster. As we previously discussed, we have not been able to simulate our base-station clustering algorithms along with our most advanced MIMO precoding schemes because of the lack of a closed form solution. Therefore, we had to resign ourselves to employing a BD solution as the one described in section 3.3.4. Even though it is not one of our particular contributions, testing our base-station clustering algorithms under such scheme is very useful because BD is probably the most commonly used linear precoding scheme for distributed multi-user MIMO systems nowadays. As for the power allocation procedure, we decided to go for uniform power allocation. As optimal power allocation only considers the inter-cluster channel matrix, that method is no longer optimal when we deal with a system formed by S clusters. Because of that, our experiments showed that the performance achieved by uniform power allocation matched that of optimal power allocation, with the optimization problem being solved by numerical methods with cvx in the latter. Therefore, given that its computational complexity is much lower, we settled for the usage of uniform power allocation.

6.2 Results

Once the simulation scenario has been thoroughly described, we will explain the particular tests we have carried out.

Under the particular conditions exposed in the previous section, $N_{\text{rep}} = 100$ sets of user locations have been generated, giving rise to $N_{\text{rep}} = 100$ different channel matrices. This can be interpreted as running our algorithms at N_{rep} unrelated time instants, in which the users being served by the cells are at distinct locations and the signal propagation conditions, represented by the Rayleigh fading, have also changed from one moment to another.

With the maximum cluster size constraints, we tried to represent a realistic situation in which the implementation cost is kept reasonably small while still obtaining a considerable increase in the system's throughput. Our main target is to compare the results achieved by our algorithms to those we can get by using the partition depicted in figure 6.3. That scheme, based on hexagonal clusters with 7 BTSs each, is very representative of the kind of partitions being used nowadays by researchers studying coordinated transmission for future mobile communication systems.

A similar cluster design but with $T = 2$ tiers would already contain 19 base-stations per cluster, probably making the implementation cost too high to be attractive for its short-term implementation. On the other hand, as we already discussed, to use any intermediate cluster size we would need to drop the hexagonal cluster shapes and use others instead. The main drawback of non-hexagonal fixed clusters is that, for a given cluster size, they have a bigger proportion of cells located in the cluster boundary. Non-fixed schemes like those obtained by our algorithms do not suffer from that because the shapes are actually chosen according to the instantaneous signal propagation conditions. However, fixed-schemes disregard that and their performance is mainly determined by the number of cells which are "protected" from interference by the cells in the cluster boundary. As a consequence, non-hexagonal fixed clusters do not exploit the increase in cluster size as much as the hexagonal ones, yielding only a moderate increase on the system's performance when $L_{\max}$ is risen. Because we wanted to give our algorithms a hard challenge by testing them against the feasible fixed scheme which maximizes the "performance-to-cluster-size-ratio", we settled for a reference partition based on hexagonal clusters with 7 BTSs.

The maximum cluster size constraints have been chosen trying to produce a fair comparison between the reference partition and those achieved by our algorithms. Furthermore, in order to explore the effects of being more or less flexible, we ended up using two different sets of maximum cluster size constraints.

The first one is completely strict as we use $L_{\max,a} = L_{\max,b} = 7$ and $P = 0$. In other words, no cluster can contain more than 7 cells. On the other hand, we also considered a set of constraints which adds some flexibility by setting $L_{\max,a} = 7$, $L_{\max,b} = 10$ and $P = 7$. Therefore, in that case, we are allowing 7 clusters to contain 8, 9 or, at most, 10 cells. The rest of clusters still must have at most 7 cells.

With the first set of maximum cluster size constraints, we are actually biasing the results in

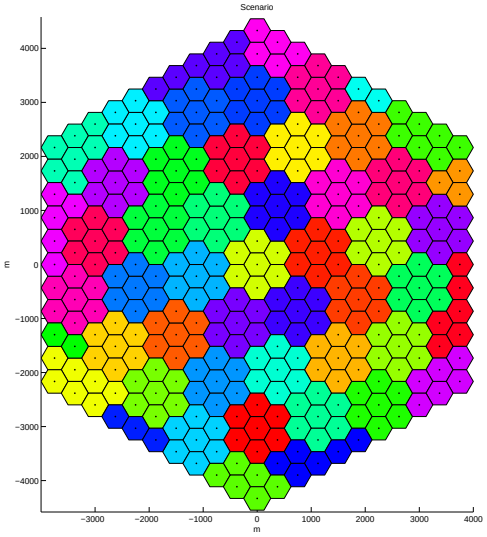


Figure 6.3: Reference partition for our simulations based on hexagonal clusters with $T = 1$ tiers. Base-stations in the outermost tiers, which cannot be assigned to hexagonal-shaped clusters due to border-effects, have been grouped by a distance criterion while keeping all the resulting clusters geographically connected.

favor of the fixed scheme of figure 6.3. That is because when we use those hexagonal shapes, the average and median cluster size is very close to 7. Indeed, the only reason why they are not exactly 7 is the existence of border effects in the outermost tiers, which do not allow to form hexagonal clusters in those areas. On the contrary, our base-station clustering algorithms converge to partitions in which many different cluster shapes may coexist, hence exhibiting a considerable variance in the cluster size. Therefore, if we force every cluster to be have no more than 7 cells, the average and median cluster sizes will be significatively lower than 7, favoring the fixed scheme based on hexagonal clusters.

Because of that, we also wanted to obtain results when we try to compensate for this effect. For such purpose, the second set of constraints was employed, as we empirically checked that under those settings, the average and median cluster sizes obtained by our base-station clustering algorithms were only slightly smaller than those of the partition shown in figure 6.3.

For every one of those different $N_{\text{rep}} = 100$ "snapshots" of the complete radio access network and for both types of constraints, we obtained a partition with each of the base-station clustering algorithms of chapter 5 under several distinct parametrization choices, which we describe next.

Spectral base-station clustering: This algorithm was tested for each of the four similarity functions described in section 5.2.1. Moreover, we also made a sweep on its parameters.

The desired number of clusters, k, was computed as $k = \text{round}\left(\alpha \frac{M}{L_{\max,a}}\right)$ with $\alpha \in [0.8, 1.4]$. Given that in our particular case $M = 331$ and $L_{\max,a} = 7$, we ended up with $k \in [38, 66]$.

Similarly, for BTS-User distance similarity and User distance similarity, the bandwidth of the Gaussian kernel, σ, was an additional parameter. For those similarity functions, we simulated bandwidths obtained as $\sigma = \beta R_{\text{cell}}$ with $\beta \in [0.5, 3]$.

Last but not least, we use algorithm 5.10 to implement step 3 in the generic base-station clustering structure 5.1. In other words, we did not use the version of the cluster splitting routine which allowed to redistribute cells with neighboring clusters when dividing a cluster. The particular cluster splitting algorithm employed within algorithm 5.10 was hierarchical divisive cluster splitting with the same similarity matrix $\mathbf{W}$ as used for the main clustering algorithm.

Hierarchical divisive spectral base-station clustering: This algorithm was also tested for all the similarity functions we proposed. Besides, a sweep on its parameters was made as well.

The maximum number of cells per cluster, $L_{\max}$, was not set fixed to neither $L_{\max,a}$ nor $L_{\max,b}$. As we already discussed, we used instead the extra degree of freedom to perform a fine tuning of the algorithm. In our simulations, we obtained $L_{\max}$ as $L_{\max} = \text{round}\,(\alpha L_{\max,a})$ with $\alpha \in [0.8, 1.4]$.

The bandwidth of the Gaussian kernel when BTS-User distance similarity or User distance similarity were used, and the particularities of the cluster splitting algorithm are setup in the same way as for spectral base-station clustering.

Hierarchical agglomerative base-station clustering: This algorithm was simulated with the same settings than hierarchical divisive spectral base-station clustering.

Mean-shift base-station clustering: The fundamental parameters of this algorithm are, as we discussed in section 5.2.5.2, the per-sample bandwidth matrices $\{\mathbf{H}_i\}_{i=1}^{M}$. In our simulations, we computed them using the K-NN heuristic described in the aforementioned section. As a recap, the idea was relatively simple. First of all, we concentrated on radially symmetric kernels so that the bandwidth matrices were actually of the form $\mathbf{H}_i = h_i^2\mathbf{I}$. In order to obtain h_i^2 for the i-th user, we computed the average distance squared to its K nearest neighbors and multiplied it by a factor α common for all users $i = 1, 2, \ldots, M$. Therefore, there are two parameters to be tested: the number of neighbors involved in the calculation, K, and the scalar factor α.

In this battery of tests, partitions were obtained for values of K ranging from 1 up to 15. The particular ranges tested for α depend on the value off K. We checked empirically that the bigger K, the lower the value of α which optimizes the algorithm's performance. As a reference, the ranges for α were chosen in such a way that for $K = 1$ we used $\alpha \in [0.4, 1.3]$ whereas for $K = 15$ we employed $\alpha \in [0.05, 0.25]$. Intermediate values of K yielded also intermediate ranges between both extremes.

We also included, as an special case, what we may denote as $K = 0$. In other words, that scenario does not employ per-samples scalar bandwidths but a simple scalar bandwidth $h_i = h \ \forall\, i = 1, \ldots, M$. In that case, the value of h was obtained as $h = \alpha R_{\text{cell}}$ with $\alpha \in [0.4, 1]$.

As for the partition refining step to ensure satisfaction of the maximum cluster size constraints, we tested both the simplified version of algorithm 5.10 and the more complex alternative 5.11 in order to test whether redistributing cells with the neighboring clusters before splitting was beneficial enough to be worth the extra complexity or not. As far as the cluster splitting algorithm is concerned, both K-means cluster splitting and hierarchical divisive cluster splitting were tested. For the latter, the similarity matrix $\mathbf{W}$ was computed using BTS-User distance similarity with $\sigma = \frac{\sqrt{3}}{2} R_{\text{cell}}$, that is, σ was chosen to equal the hexagon's apothem.

Genetic mean-shift base-station clustering: Most parameters of this algorithm were determined by the need of keeping the computational complexity at reasonable levels.

A moderate population size of $m = 50$ was used, with a mating pool of size $e = 46$ which allowed us to employ elitism with the four fittest individuals of each generation. The mutation probability was chosen to be $p_m = 0.02$ whereas the crossover probability was set to $p_c = 0.75$. With those settings, our experiments seemed to confirm that a proper balance between exploration and exploitation was achieved. On the other hand, two different fitness functions were considered: the system's sum-rate and median rate.

The number of generations to be simulated was limited to $T = 125$, mainly due to computational considerations. However, as we will discuss later, that number of iterations was enough to get very close to convergence.

To add even more genetic diversity in the population, the particular algorithm used for the partition refining step to ensure satisfaction of the maximum cluster size constraints was chosen randomly. In our implementation, algorithm 5.10 was to be employed with probability 0.8 whereas its more complex counterpart, algorithm 5.11, had an associated probability of 0.2. The three different types of cluster splitting algorithms available were also picked up randomly, in this case each with a probability of 1/3.

Evolutionary base-station clustering: The parametrization of our final algorithm is analogous to that of genetic mean-shift base-station clustering, being the only difference that in this case we do not have crossover at all and mutation occurs always with probability 1, as we discussed in section 5.2.7.

Even though we have discussed the ranges we have tested for the parameters governing the behavior of our algorithms, we still have to define properly what we will understand by "optimizing" or "tuning" the parameters of a base-station clustering algorithm.

At first thought, the definition can appear to be trivial: the particular choice of parameters which maximizes whatever performance measure we consider, in our case that would be the sum-rate or median-rate of the system, is the optimal and the performance they achieve is the one taken for evaluating the algorithm.

However, there is a subtle yet fundamental issue which the previous definition disregards: the scope in which the optimization process is carried out. In this way, we considered two different approaches:

Dynamic parameter tuning: In this case, the algorithm's parameters are adjusted independently for each particular scenario. This allows the algorithm to adapt much better to the specific signal propagation conditions at each particular instant, greatly enhancing the final performance.

However, the computational complexity required to make a complete sweep on all the parameters for each execution of the algorithm is too high. As a consequence, this way of tuning the parameters is unfeasible nowadays for implementation in a real mobile communications network. Therefore, just like our evolutionary base-station clustering algorithms, their usefulness resides in their value for researching and designing other base-station clustering algorithms.

Static parameter tuning: The alternative to the previous scheme is to carry out an offline tuning of the parameters. In this case, we will average the results for a big number of different signal propagation conditions and user locations, in our case, $N_{\text{rep}} = 100$. Then, the set of parameters which optimizes the averaged performance achieved in all the N_{rep} scenarios is considered to be the optimal. For instance, a particular implementation of this idea could use a set of "snapshots" of the system conditions obtained during a complete day as a data base to tune the parameters of the base-station clustering algorithm which is to be employed in the following day. In this way, the tuning process would be executed just once per day, instead of each time the base-station clustering algorithm is to be used.

The fundamental advantage of this scheme is that, as the parameter tuning process is not done during the normal operation of the network, the computational complexity is greatly reduced. However, as the reader may imagine, we lose a great amount of flexibility, significatively degrading the overall performance increase with respect to the reference partition based on fixed hexagonal clusters.

We will now show both in tabular and graphical formats the results obtained in the simulation scenario previously described. Eight different tables are included, each representing different conditions of the simulation. They are organized as follows.

The first four tables show the results achieved when we use the restricted set of constraints, that is, $L_{\max,a} = L_{\max,b} = 7$ and $P = 0$. On the contrary, the last four tables include the results obtained for the flexible set of constraints, which consisted of $L_{\max,a} = 7$, $L_{\max,b} = 10$

and $P = 7$. Each of the previous groups of tables are subdivided again in two. The first two tables of each group show results obtained while optimizing the system's sum rate, whereas the last two tables are achieved when optimizing the system's median rate. Finally, every one of the previous pairs of tables contains then results when dynamic and static parameter tuning are used respectively. Note that, for static parameter tuning, the two base-station clustering algorithms based on evolutionary computation are not included, as they have a inherently dynamic nature.

In each table, we depict by rows the results achieved by each particular algorithm. Moreover, most algorithms have been executed under slightly different conditions which are also listed in the table. For reasons of space, the names of the algorithms have been shortened using acronyms which are defined at the beginning of this document even though we believe that they are still clearly recognizable.

In each column we show the value of a magnitude of interest. All the results shown are the arithmetic average of the values for each magnitude achieved in each of the $N_{\text{rep}} = 100$ different scenarios we have simulated. The first two columns refer to the system's average rate per user, $\overline{R_i}$, and the system's median rate per user, $\widetilde{R_i}$, respectively. The other 4 columns show figures relative to the cluster sizes. The third column shows the average cluster size, $\overline{L_i}$. The forth column shows the maximum cluster size in the partition, $\max(L_i)$. The fifth one, the median cluster size, $\widetilde{L_i}$. And, finally, the last column shows the number of clusters in the partition which surpassed the lower maximum cluster size threshold $L_{\max,a}$ which, in our case, was set to 7 cells.

General discussion on the results

First of all, grouping base-stations in clusters to carry out coordinated transmission within the cluster provides a remarkable increase in the system throughput, even when the clusters are made with a fixed scheme like the one of figure 6.3. The increase in the system's average rate is quite high. Still, the rise in the median rate is much more spectacular, enhancing it by approximately 17%. Therefore, our simulations confirm that next generation mobile communications will need to use coordinated MIMO precoding or filtering in clusters in order to satisfy the ambitious throughput specifications expected of those systems.

Nonetheless, probably the most important figures are those referring to the performance achieved by genetic mean-shift clustering and evolutionary base-station clustering. The average rate and especially the median rate they attain are simply astonishing, exceeding by a considerable margin our expectations. As we can see in the tables, those two algorithms have allowed us to prove empirically that there exist partitions able to enhance the system median rate by as much as 37% respect to the system without coordination and close to 17% with respect to the scheme with fixed hexagonal clusters of 7 base-stations. Not only that, they even achieve that using a significatively smaller average and median cluster size than the aforementioned fixed scheme, in the order of one cell less per cluster.

In short, thanks to those algorithms, we can claim that we have provided with enough empirical evidence to justify opening new research lines for developing better and faster clustering

SCHEMES		$\bar{R}_i$	$\tilde{R}_i$	$\bar{L}_i$	$\max(L_i)$	$\tilde{L}_i$	$\sum_i [L_i > L_{\max,a}]$
Uncoordinated		5,4370	3,9450	1,0000	1,0000	1,0000	0,0000
Non-adaptive hexagonal clustering		5,9014	4,6149	6,4902	7,0000	7,0000	0,0000
SC	Unnormalized interference similarity	5,9189	4,6748	5,6252	7,0000	5,9200	0,0000
	Normalized interference similarity	5,8062	4,5800	5,7347	7,0000	6,0300	0,0000
	BTS-user distance similarity	6,0232	4,7595	5,6099	7,0000	5,7700	0,0000
	User distance similarity	5,9712	4,5341	5,6500	7,0000	5,8750	0,0000
HDSC	Unnormalized interference similarity	5,8930	4,4204	5,6005	7,0000	5,9219	0,0000
	Normalized interference similarity	5,7842	4,3769	5,6394	7,0000	5,8750	0,0000
	BTS-user distance similarity	5,9678	4,4966	5,5374	7,0000	5,6875	0,0000
	User distance similarity	5,9442	4,5085	5,5795	7,0000	5,8021	0,0000
HAC	Unnormalized interference similarity	5,8801	4,4197	5,4906	7,0000	5,7031	0,0000
	Normalized interference similarity	5,7997	4,3032	5,5674	7,0000	5,9479	0,0000
	BTS-user distance similarity	5,9671	4,5073	5,6346	7,0000	5,9635	0,0000
	User distance similarity	5,9329	4,4893	5,7082	7,0000	6,1146	0,0000
MS-BTS-C	K-means cluster splitting	5,9849	4,7319	5,4180	7,0000	5,2750	0,0000
	K-means neighbors cluster splitting	6,0393	4,7415	6,0091	7,0000	6,0600	0,0000
	Hierarchical cluster splitting	5,9913	4,7433	5,5208	7,0000	5,5850	0,0000
	Mean shift neighbors cluster splitting	5,9925	4,7446	5,5127	7,0000	5,5850	0,0000
GEN-BTS-C		6,2165	4,9876	5,9758	7,0000	6,2350	0,0000
EVOL-MS-BTS-C		6,2310	4,9822	5,6260	7,0000	5,8200	0,0000

Table 6.1: Results obtained for the strict set of constraints with dynamic parameter tuning trying to optimize the sum-rate.

SCHEMES		$\bar{R}_i$	$\tilde{R}_i$	$\bar{L}_i$	$\max(L_i)$	$\tilde{L}_i$	$\sum_i [L_i > L_{\max,a}]$
Uncoordinated		5,4370	3,9450	1,0000	1,0000	1,0000	0,0000
Non-adaptive hexagonal clustering		5,9014	4,6149	6,4902	7,0000	7,0000	0,0000
SC	Unnormalized interference similarity	5,8543	4,6008	5,5709	7,0000	5,9000	0,0000
	Normalized interference similarity	5,7472	4,5397	5,7060	7,0000	6,0250	0,0000
	BTS-user distance similarity	5,9034	4,6590	5,6398	7,0000	5,8800	0,0000
	User distance similarity	5,8677	4,6270	5,5624	7,0000	5,7500	0,0000
HDSC	Unnormalized interference similarity	5,8544	4,5839	5,5887	7,0000	5,9167	0,0000
	Normalized interference similarity	5,7495	4,5345	5,6418	7,0000	5,9531	0,0000
	BTS-user distance similarity	5,8721	4,6178	5,5517	7,0000	5,8333	0,0000
	User distance similarity	5,8462	4,6258	5,5894	7,0000	5,8490	0,0000
HAC	Unnormalized interference similarity	5,8411	4,5892	5,5273	7,0000	5,7188	0,0000
	Normalized interference similarity	5,7558	4,4646	5,5315	7,0000	5,8125	0,0000
	BTS-user distance similarity	5,8528	4,5862	5,6797	7,0000	6,0000	0,0000
	User distance similarity	5,8239	4,5884	5,4803	7,0000	5,7500	0,0000
MS-BTS-C	K-means cluster splitting	5,9008	4,6746	5,3620	7,0000	5,1700	0,0000
	K-means neighbors cluster splitting	5,9149	4,6801	5,6357	7,0000	5,7050	0,0000
	Hierarchical cluster splitting	5,9093	4,6823	5,4526	7,0000	5,3850	0,0000
	Mean shift neighbors cluster splitting	5,9038	4,6863	5,4507	7,0000	5,3750	0,0000

Table 6.2: Results obtained for the strict set of constraints with static parameter tuning trying to optimize the sum-rate.

SCHEMES		$\bar{R}_i$	$\tilde{R}_i$	$\bar{L}_i$	$\max(L_i)$	$\tilde{L}_i$	$\sum_i [L_i > L_{\max,a}]$
Uncoordinated		5,4370	3,9450	1,0000	1,0000	1,0000	0,0000
Non-adaptive hexagonal clustering		5,9014	4,6149	6,4902	7,0000	7,0000	0,0000
SC	Unnormalized interference similarity	5,8751	4,7720	5,6049	7,0000	5,8450	0,0000
	Normalized interference similarity	5,7641	4,6680	5,6909	7,0000	5,9800	0,0000
	BTS-user distance similarity	5,9405	4,9570	5,5920	7,0000	5,7000	0,0000
	User distance similarity	5,6597	4,8907	5,4080	6,7200	5,6450	0,0000
HDSC	Unnormalized interference similarity	5,6234	4,6867	5,3894	6,7200	5,6550	0,0000
	Normalized interference similarity	5,5269	4,6287	5,3982	6,7200	5,6800	0,0000
	BTS-user distance similarity	5,6649	4,8568	5,3014	6,7200	5,3800	0,0000
	User distance similarity	5,6331	4,8550	5,3354	6,7200	5,5600	0,0000
HAC	Unnormalized interference similarity	5,6078	4,6951	5,2317	6,7200	5,4200	0,0000
	Normalized interference similarity	5,5355	4,5730	5,3479	6,7200	5,6300	0,0000
	BTS-user distance similarity	5,6575	4,8638	5,3718	6,7200	5,6700	0,0000
	User distance similarity	5,6156	4,8440	5,4236	6,7200	5,8750	0,0000
MS-BTS-C	K-means cluster splitting	5,9112	4,8907	5,3798	7,0000	5,2650	0,0000
	K-means neighbors cluster splitting	5,9507	4,9466	5,8659	7,0100	5,9150	0,0100
	Hierarchical cluster splitting	5,9115	4,8980	5,4763	7,0000	5,4900	0,0000
	Mean shift neighbors cluster splitting	5,9194	4,9196	5,4781	7,0000	5,6200	0,0000
GEN-BTS-C		6,0082	5,3290	5,8927	7,0000	6,1050	0,0000
EVOL-MS-BTS-C		5,9981	5,3830	5,5519	7,0000	5,5800	0,0000

Table 6.3: Results obtained for the strict set of constraints with dynamic parameter tuning trying to optimize the median-rate.

SCHEMES		$\bar{R}_i$	$\tilde{R}_i$	$\bar{L}_i$	$\max(L_i)$	$\tilde{L}_i$	$\sum_i [L_i > L_{\max,a}]$
Uncoordinated		5,4370	3,9450	1,0000	1,0000	1,0000	0,0000
Non-adaptive hexagonal clustering		5,9014	4,6149	6,4902	7,0000	7,0000	0,0000
SC	Unnormalized interference similarity	5,8490	4,6017	5,5858	7,0000	5,8600	0,0000
	Normalized interference similarity	5,7423	4,5607	5,6350	7,0000	5,9650	0,0000
	BTS-user distance similarity	5,9002	4,6825	5,4867	7,0000	5,5400	0,0000
	User distance similarity	5,8591	4,6497	5,5380	7,0000	5,8021	0,0000
HDSC	Unnormalized interference similarity	5,8544	4,5839	5,5887	7,0000	5,9167	0,0000
	Normalized interference similarity	5,7470	4,5469	5,6381	7,0000	5,9167	0,0000
	BTS-user distance similarity	5,8540	4,6487	5,6009	7,0000	5,8750	0,0000
	User distance similarity	5,8441	4,6267	5,6148	7,0000	5,9219	0,0000
HAC	Unnormalized interference similarity	5,8321	4,5916	5,4187	7,0000	5,4948	0,0000
	Normalized interference similarity	5,7539	4,4804	5,5894	7,0000	5,8906	0,0000
	BTS-user distance similarity	5,8403	4,6332	5,4614	7,0000	5,5052	0,0000
	User distance similarity	5,8121	4,6199	5,5942	7,0000	5,9375	0,0000
MS-BTS-C	K-means cluster splitting	5,8956	4,6786	5,3304	7,0000	5,0850	0,0000
	K-means neighbors cluster splitting	5,9089	4,6801	5,6539	7,0000	5,7650	0,0000
	Hierarchical cluster splitting	5,9024	4,6845	5,3866	7,0000	5,2600	0,0000
	Mean shift neighbors cluster splitting	5,9001	4,6910	5,4372	7,0000	5,4000	0,0000

Table 6.4: Results obtained for the strict set of constraints with static parameter tuning trying to optimize the median-rate.

SCHEMES		$\bar{R}_i$	$\tilde{R}_i$	$\bar{L}_i$	$\max(L_i)$	$\tilde{L}_i$	$\sum_i [L_i > L_{\max,a}]$
Uncoordinated		5,4370	3,9450	1,0000	1,0000	1,0000	0,0000
Non-adaptive hexagonal clustering		5,9014	4,6149	6,4902	7,0000	7,0000	0,0000
SC	Unnormalized interference similarity	5,9238	4,6511	5,9232	9,3000	6,0650	6,7500
	Normalized interference similarity	5,8083	4,5769	5,9628	8,9900	6,1900	6,8400
	BTS-user distance similarity	6,0367	4,7826	5,9806	8,6100	6,0650	6,0000
	User distance similarity	5,9788	4,5836	5,9370	8,5567	6,0258	6,3402
HDSC	Unnormalized interference similarity	5,8913	4,4641	5,7769	7,8144	6,0052	4,4124
	Normalized interference similarity	5,7829	4,4302	5,7486	7,5670	5,9227	3,2990
	BTS-user distance similarity	5,9729	4,5777	5,8430	8,0206	5,8969	5,2268
	User distance similarity	5,9488	4,5719	5,8391	8,0619	5,9072	5,5258
HAC	Unnormalized interference similarity	5,8692	4,4589	5,7334	8,5464	5,7990	5,1959
	Normalized interference similarity	5,7880	4,3364	5,7373	8,3505	6,0773	4,8351
	BTS-user distance similarity	5,9644	4,5680	5,8771	8,4536	6,1031	4,9072
	User distance similarity	5,9310	4,5166	5,8988	8,4330	6,1289	4,4021
MS-BTS-C	K-means cluster splitting	6,0026	4,7472	5,6773	8,4200	5,5900	4,2000
	K-means neighbors cluster splitting	6,0434	4,7731	6,1864	8,7500	6,2100	6,0800
	Hierarchical cluster splitting	6,0020	4,7478	5,6945	8,4400	5,6800	3,9800
	Mean shift neighbors cluster splitting	6,0037	4,7708	5,7173	8,8800	5,7100	3,7600
GEN-BTS-C		6,2304	4,9857	6,1727	9,0543	6,4239	3,9239
EVOL-MS-BTS-C		6,2643	5,0276	5,8331	9,0652	5,8750	4,3587

Table 6.5: Results obtained for the flexible set of constraints with dynamic parameter tuning trying to optimize the sum-rate.

SCHEMES		$\bar{R}_i$	$\tilde{R}_i$	$\bar{L}_i$	$\max(L_i)$	$\tilde{L}_i$	$\sum_i [L_i > L_{\max,a}]$
Uncoordinated		5,4370	3,9450	1,0000	1,0000	1,0000	0,0000
Non-adaptive hexagonal clustering		5,9014	4,6149	6,4902	7,0000	7,0000	0,0000
SC	Unnormalized interference similarity	5,8535	4,5793	5,9216	9,3800	6,0500	6,6600
	Normalized interference similarity	5,7485	4,5215	5,9845	9,0900	6,2750	6,9300
	BTS-user distance similarity	5,9207	4,6812	6,0604	8,6600	6,0250	6,9400
	User distance similarity	5,8685	4,6483	5,9222	8,8454	6,0052	6,5876
HDSC	Unnormalized interference similarity	5,8461	4,5683	5,8232	8,3299	5,9897	7,0000
	Normalized interference similarity	5,7436	4,5315	5,6424	7,0000	5,9536	0,0000
	BTS-user distance similarity	5,8734	4,6405	5,8849	8,3505	5,9948	7,0000
	User distance similarity	5,8429	4,6088	5,9398	8,3608	6,0309	7,0000
HAC	Unnormalized interference similarity	5,8238	4,5749	5,9472	10,0000	5,8763	7,0000
	Normalized interference similarity	5,7390	4,4480	5,9327	10,0000	6,0309	7,0000
	BTS-user distance similarity	5,8457	4,5810	5,8135	8,0000	6,1753	7,0000
	User distance similarity	5,8107	4,6029	5,7408	8,0000	6,0000	7,0000
MS-BTS-C	K-means cluster splitting	5,9130	4,6735	5,7404	9,1000	5,7200	5,6300
	K-means neighbors cluster splitting	5,9245	4,6392	6,3724	8,8900	6,3650	6,5100
	Hierarchical cluster splitting	5,9114	4,6803	5,7512	9,1100	5,7600	5,6000
	Mean shift neighbors cluster splitting	5,9084	4,6617	5,6372	8,9100	5,5450	4,2100

Table 6.6: Results obtained for the flexible set of constraints with static parameter tuning trying to optimize the sum-rate.

SCHEMES		$\bar{R}_i$	$\tilde{R}_i$	$\bar{L}_i$	$\max(L_i)$	$\tilde{L}_i$	$\sum_i [L_i > L_{\max,a}]$
Uncoordinated		5,4370	3,9450	1,0000	1,0000	1,0000	0,0000
Non-adaptive hexagonal clustering		5,9014	4,6149	6,4902	7,0000	7,0000	0,0000
SC	Unnormalized interference similarity	5,8735	4,7630	5,9336	9,2100	6,0700	6,8000
	Normalized interference similarity	5,7623	4,6877	5,9580	8,9500	6,1750	6,8800
	BTS-user distance similarity	5,9467	4,9700	5,8937	8,3600	5,9150	5,9600
	User distance similarity	5,7241	4,9042	5,7315	8,3400	5,8300	6,1400
HDSC	Unnormalized interference similarity	5,6731	4,6847	5,5582	7,4600	5,7550	3,6400
	Normalized interference similarity	5,5796	4,6376	5,5633	7,3000	5,8400	2,8600
	BTS-user distance similarity	5,7208	4,8794	5,5814	7,5700	5,6500	4,3500
	User distance similarity	5,6990	4,8680	5,5827	7,5300	5,6400	4,2100
HAC	Unnormalized interference similarity	5,6564	4,6783	5,5509	8,3400	5,6450	5,1800
	Normalized interference similarity	5,5807	4,5624	5,5886	8,2500	5,8200	5,1100
	BTS-user distance similarity	5,7126	4,8690	5,6064	8,0700	5,8250	4,7600
	User distance similarity	5,6732	4,8413	5,6722	8,1000	5,9950	4,6200
MS-BTS-C	K-means cluster splitting	5,9327	4,9037	5,5731	8,1700	5,4800	3,0500
	K-means neighbors cluster splitting	5,9560	4,9408	5,9604	8,5800	5,9400	4,9500
	Hierarchical cluster splitting	5,9316	4,9048	5,6452	8,3100	5,5900	3,7000
	Mean shift neighbors cluster splitting	5,9321	4,9129	5,6537	8,5100	5,6500	3,2000
GEN-BTS-C		6,0255	5,3494	6,0607	8,7500	6,2550	3,2600
EVOL-MS-BTS-C		6,0031	5,3978	5,7369	8,6100	5,7900	2,8900

Table 6.7: Results obtained for the flexible set of constraints with dynamic parameter tuning trying to optimize the median-rate.

SCHEMES		$\bar{R}_i$	$\tilde{R}_i$	$\bar{L}_i$	$\max(L_i)$	$\tilde{L}_i$	$\sum_i [L_i > L_{\max,a}]$
Uncoordinated		5,4370	3,9450	1,0000	1,0000	1,0000	0,0000
Non-adaptive hexagonal clustering		5,9014	4,6149	6,4902	7,0000	7,0000	0,0000
SC	Unnormalized interference similarity	5,8493	4,5951	5,9209	9,2800	6,0550	6,8900
	Normalized interference similarity	5,7406	4,5581	5,9353	9,1500	6,0850	6,9600
	BTS-user distance similarity	5,9066	4,6917	6,0958	8,3600	6,0750	6,9800
	User distance similarity	5,8534	4,6556	5,8213	9,1237	5,9227	5,8247
HDSC	Unnormalized interference similarity	5,8461	4,5683	5,8232	8,3299	5,9897	7,0000
	Normalized interference similarity	5,7394	4,5468	5,8720	8,0000	6,0773	6,8969
	BTS-user distance similarity	5,8614	4,6560	5,8606	8,3505	5,9948	7,0000
	User distance similarity	5,8375	4,6294	5,9044	8,4124	6,0773	7,0000
HAC	Unnormalized interference similarity	5,8165	4,5850	5,7343	9,0000	5,6907	7,0000
	Normalized interference similarity	5,7390	4,4480	5,9327	10,0000	6,0309	7,0000
	BTS-user distance similarity	5,8337	4,6216	5,9003	9,9691	5,9433	7,0000
	User distance similarity	5,7990	4,6135	5,8279	7,0000	6,8969	0,0000
MS-BTS-C	K-means cluster splitting	5,9024	4,6887	5,5229	8,5100	5,3550	2,7600
	K-means neighbors cluster splitting	5,9050	4,6906	5,6063	8,8300	5,4800	3,9100
	Hierarchical cluster splitting	5,9040	4,6899	5,5916	8,8300	5,4500	3,8800
	Mean shift neighbors cluster splitting	5,9031	4,6881	5,4586	8,1900	5,3200	1,6600

Table 6.8: Results obtained for the flexible set of constraints with static parameter tuning trying to optimize the median-rate.

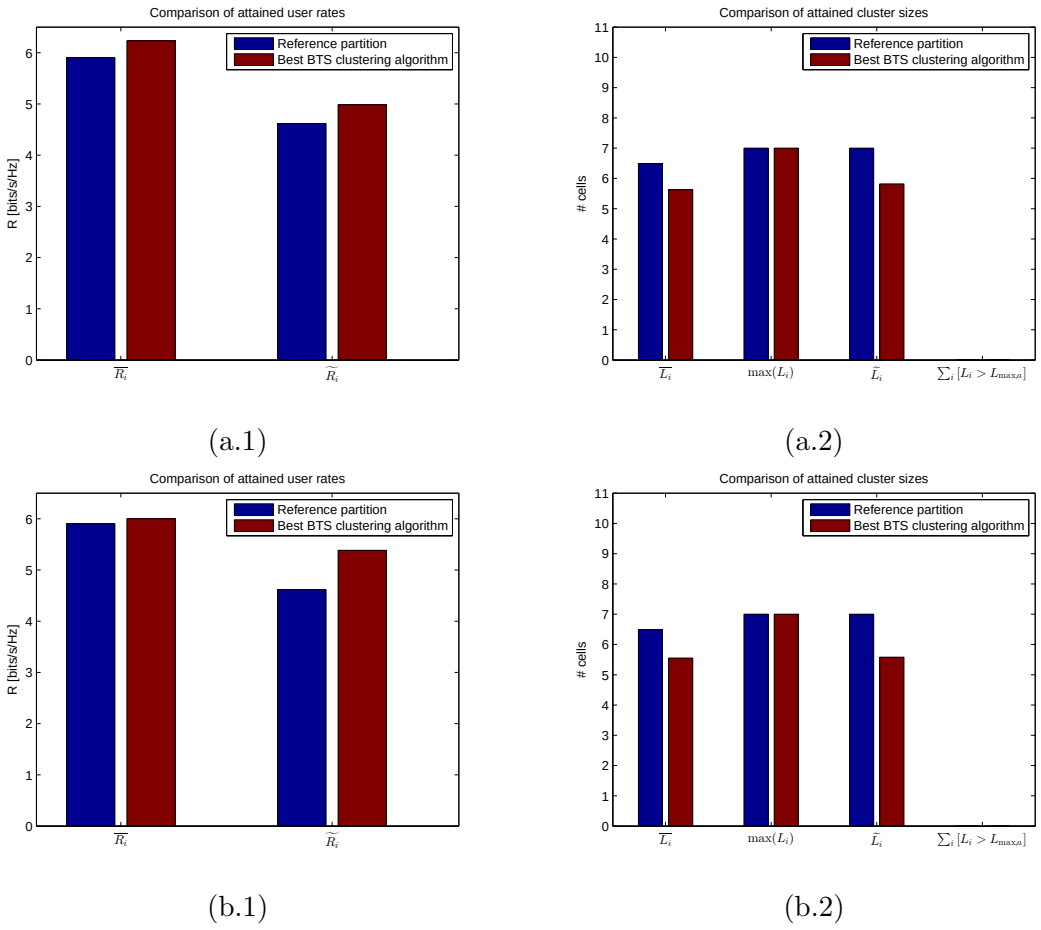


Figure 6.4: Comparison of results attained by reference partition 6.3 and the base-station clustering algorithm which performed better under the strict set of constraints with dynamic parameter tuning. In (a) we show results attained when optimizing the average-rate whereas in (b) we optimized the median-rate.

algorithms which dynamically obtain partitions of the set of base-stations within the mobile communications network. Given the current need in those networks for enhancing the spectral efficiency as much as possible, the performance increase that can be obtained by carefully designing the clusters is just too big to be disregarded.

Another interesting observation can be induced from the results of those algorithms. As we can see, there is a significant increase both when the sum-rate and median-rate are to be optimized. However, the performance gap achieved when trying to improve the median-rate is our main goal is much bigger than the one attained when optimizing the average rate. Not only that, if we look at the complete table, we can see that the previous statement also holds true for absolutely every one of our non-evolutionary base-station clustering algorithms, from spectral base-station clustering up to mean-shift base-station clustering. No matter the specific algorithm, the choice for the similarity function or the way of splitting the clusters violating the size constraints, in any case, the edge obtained when optimizing the median rate is much more

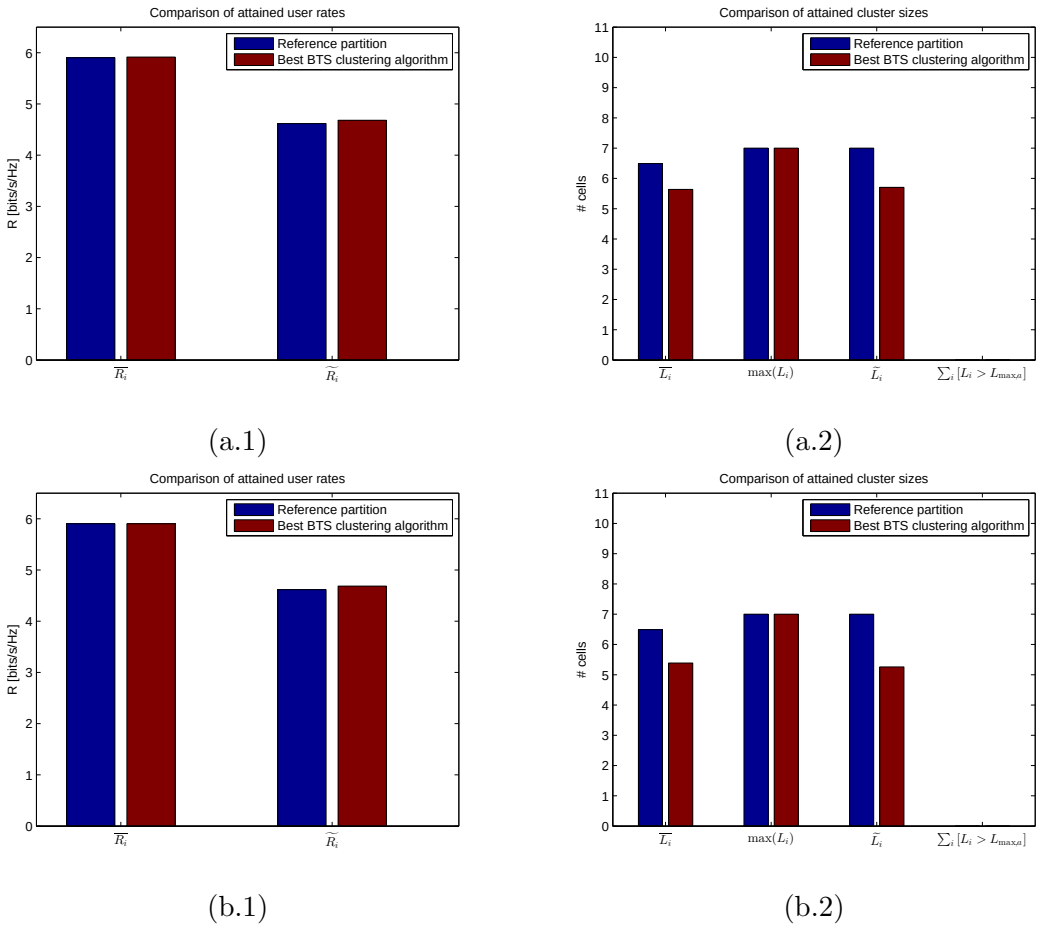


Figure 6.5: Comparison of results attained by reference partition 6.3 and the base-station clustering algorithm which performed better under the strict set of constraints with static parameter tuning. In (a) we show results attained when optimizing the average-rate whereas in (b) we optimized the median-rate.

significant.

At first thought, it may seem that the reason for that lies within our algorithms. However, if we think about it carefully, we can conclude that it is not that our algorithms perform better when they are tuned to optimize the median rate instead of the average rate. On the contrary, what is really happening is that the fixed scheme with hexagonal clusters of figure 6.3 is implicitly thought to optimize the sum-rate, hence the potential improvement margin for the average rate becomes smaller. Why is that so? Recall that the basic idea we argued to motivate any fixed partition was nothing but sacrificing some unlucky cells by placing them in the cluster boundary, so that they can act as a shield against interference for the rest of cells in the cluster. In this way, hexagonal clusters were optimal because, for a fixed cluster size, they minimize the number of cells in the boundary. However, this motivation is also the reason why they exhibit a behavior biased towards sum-rate maximization. They are an "elitist" approach in which some cells in each cluster are left out to have a terrible performance in purpose, in order to have some

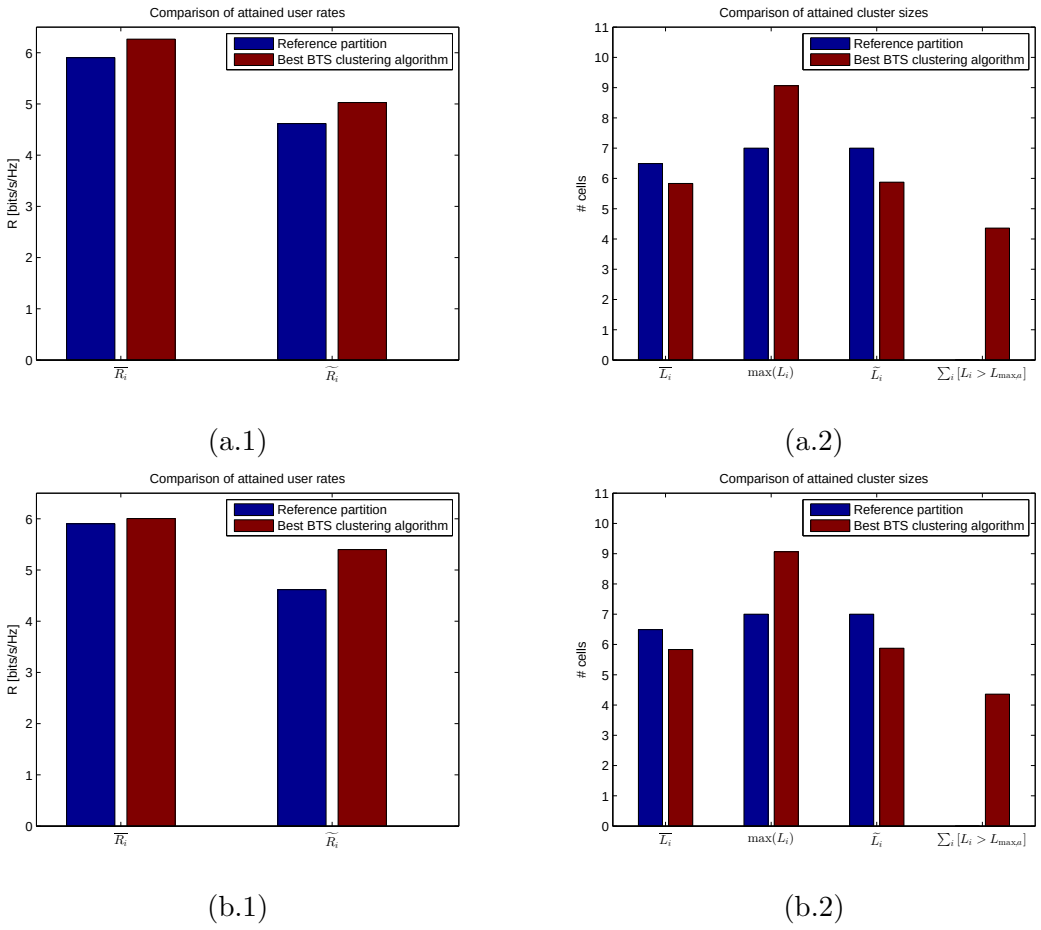


Figure 6.6: Comparison of results attained by reference partition 6.3 and the base-station clustering algorithm which performed better under the flexible set of constraints with dynamic parameter tuning. In (a) we show results attained when optimizing the average-rate whereas in (b) we optimized the median-rate.

other cells in the same cluster which have very low interference levels and achieve a quite high throughput. Clearly, the median rate suffers in this situation but the inner cells of each cluster act as outliers with a high rate, thus significatively increasing the average rate of the system. In any case, no matter which is the objective function to be maximized, there is a considerable margin for improvement which our algorithms are able to exploit, as the results reflect. To make things even better, we can also see in the tables and bar charts that, no matter what we try to optimize, the other performance measures are also enhanced. For instance, when genetic mean-shift base-station clustering is set to work with sum-rate fitness, the median rate increases almost by 10% with respect to the fixed partition, even if that was not the algorithm's main goal. In that sense, average-rate maximization tends to be more balanced whereas median-rate maximization greatly enhances the median-rate but barely increases the average rate. This behavior also occurs in the non-evolutionary algorithms, but in a less pronounced way.

Because the partitions achieved by the two evolutionary algorithms represent quasi-optimal

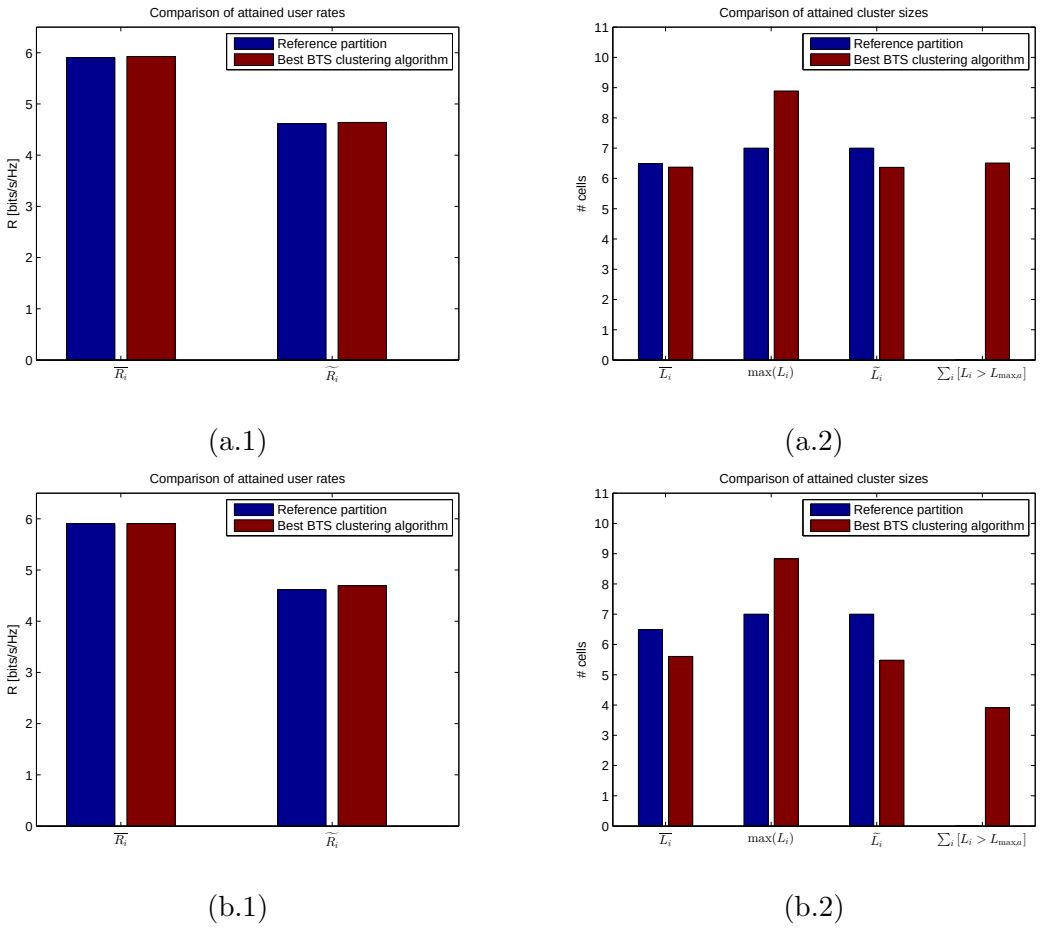


Figure 6.7: Comparison of results attained by reference partition 6.3 and the base-station clustering algorithm which performed better under the flexible set of constraints with static parameter tuning. In (a) we show results attained when optimizing the average-rate whereas in (b) we optimized the median-rate.

solutions, we believe that they have a special value from a theoretical point of view and deserve further study. As extra reference, we provide in figures 6.8 and 6.9 the evolution of the fitness achieved by those two algorithms as the generations go by. The first figure refers to the results obtained with the strict set of maximum cluster size constraints, whereas the second figure was obtained while employing the flexible set of constraints. On the other hand, we also depict in figures 6.10 and 6.11 the CDF of the user rates attained by both aforementioned algorithms. The plots include results obtained with both strict and flexible sets of constraints as well as sum-rate and median-rate fitnesses.

There are several fundamental observations which can be extracted from those pictures.

The first thing which we notice when looking at them is the huge performance gap we were talking about before: in all cases, the fittest individual of the last generation has a much better performance that the fixed scheme. However, there are also several things illustrated in those figures which cannot be appreciated from the information contained in the tables and bar charts.

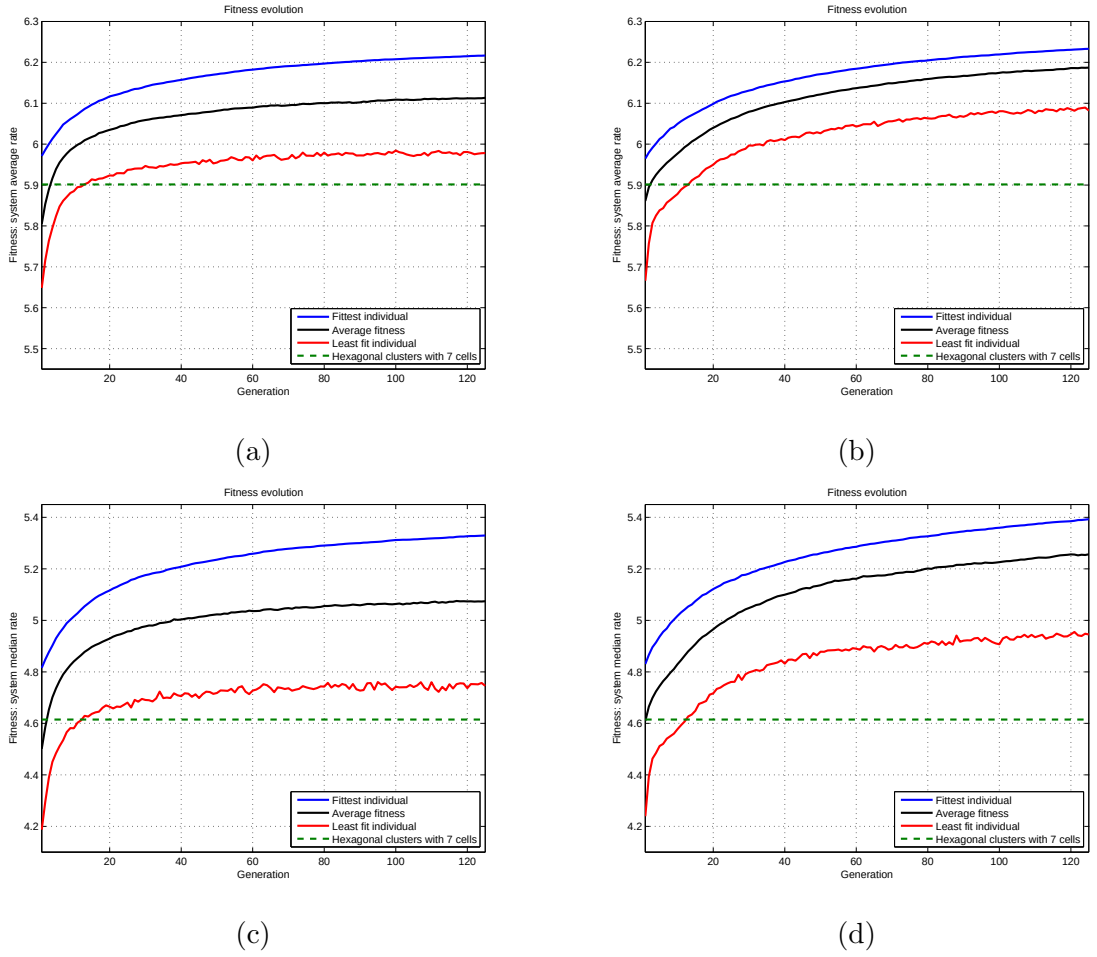


Figure 6.8: Evolution of fitness for evolutionary base-station clustering algorithms with the strict set of maximum cluster size constraints. In (a) we show the results achieved by evolutionary base-station clustering with sum-rate fitness. In (b) we show the results achieved by genetic mean-shift base-station clustering with sum-rate fitness. In (c) we show the results achieved by evolutionary base-station clustering with median-rate fitness. In (d) we show the results achieved by genetic mean-shift base-station clustering with median-rate fitness.

On the one hand, we see that the evolution process happens quite fast. In barely 20 generations, both genetic mean-shift clustering and evolutionary base-station clustering have already progressed enormously. After around 100 generations, the slope becomes very small indicating that we are pretty close to convergence. Looking at the trend, probably 150 iterations, 200 at most, would suffice to converge to the optimal partition. Being pragmatic, the performance increase which could be achieved by rising the number of generations to be simulated from 100 to 200 is most likely not even worth duplicating the computational effort. This observation is important, since it encourages us to believe that there must exist a fast, efficient and, most importantly, feasible way of obtaining suboptimal partitions which still exhibit a huge

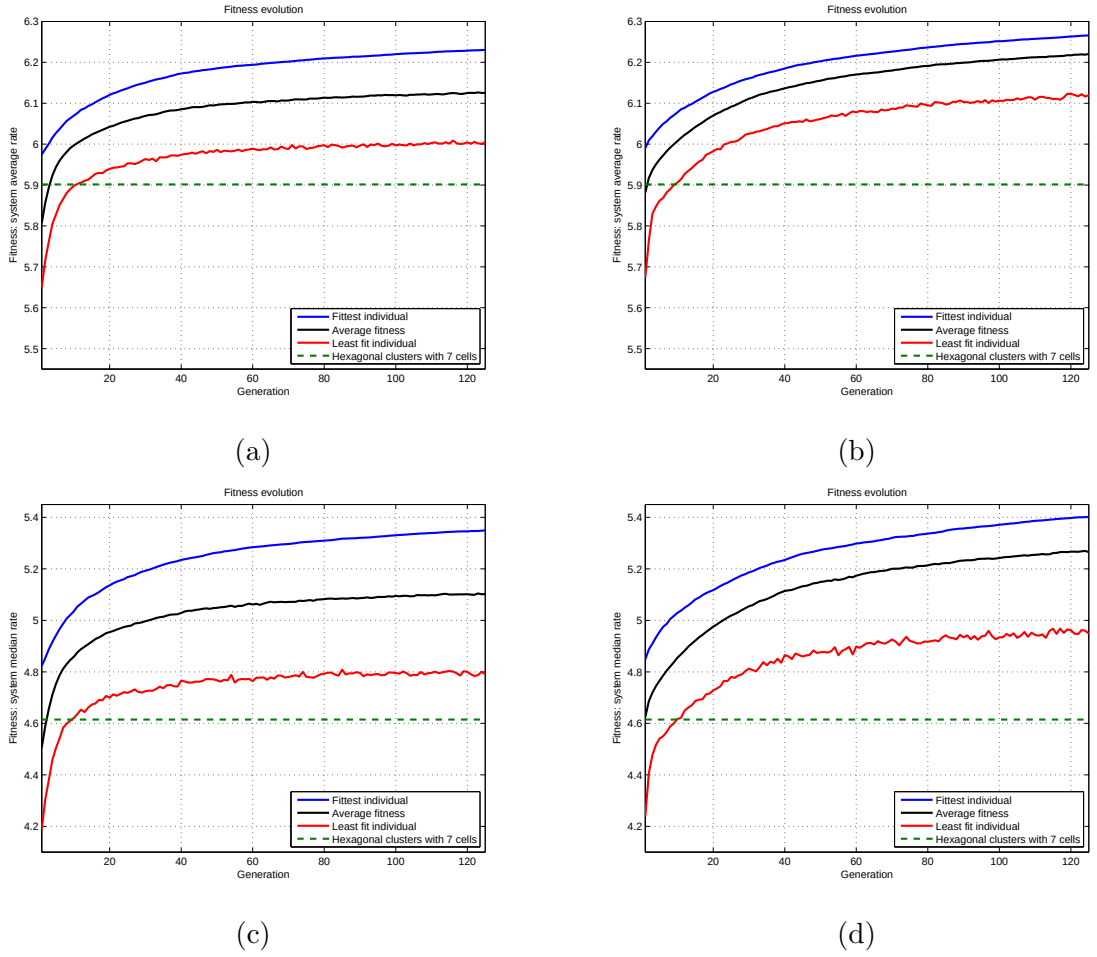
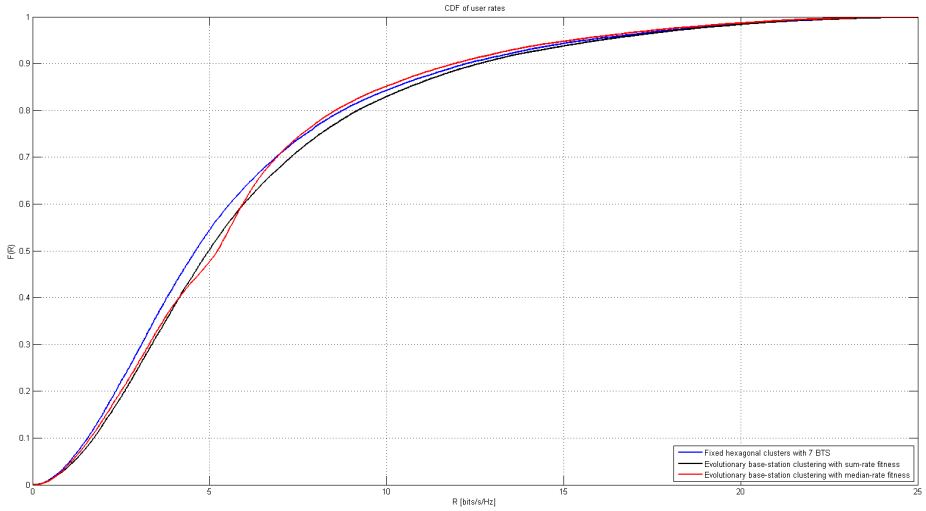


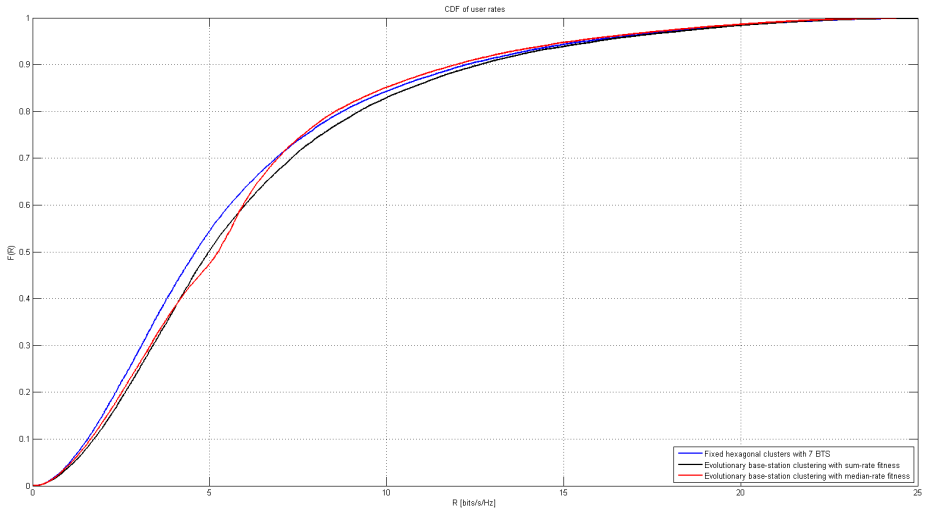
Figure 6.9: Evolution of fitness for evolutionary base-station clustering algorithms with the flexible set of maximum cluster size constraints. In (a) we show the results achieved by evolutionary base-station clustering with sum-rate fitness. In (b) we show the results achieved by genetic mean-shift base-station clustering with sum-rate fitness. In (c) we show the results achieved by evolutionary base-station clustering with median-rate fitness. In (d) we show the results achieved by genetic mean-shift base-station clustering with median-rate fitness.

performance increase with respect to any fixed scheme. Finding them represents an interesting challenge which could provide an edge in future mobile communications systems.

On the other hand, it is a bit surprising for us that the partitions obtained with the flexible set of constraints are barely better than those obtained with the strict set of constraints. This is again a wonderful result since it shows that, even under very stringent conditions which do not leave many degrees of freedom, it is still possible to do very well. According to our simulations, we see that allowing around 12.5% of the clusters to have up to 3 extra cells is not something determinant. The performance does increase a little but not enough to justify the extra implementation cost. This is not a unique trait of the evolutionary algorithms. If we



(a) Strict set of constraints

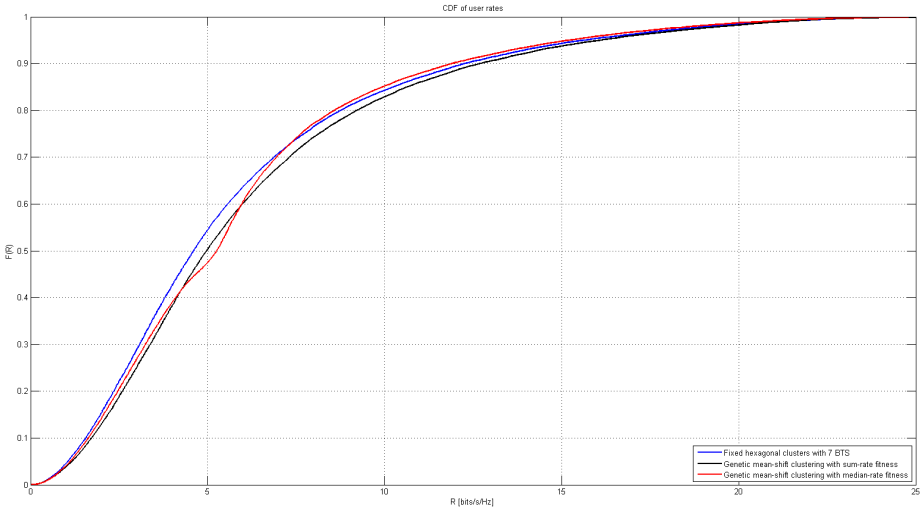


(b) Flexible set of constraints

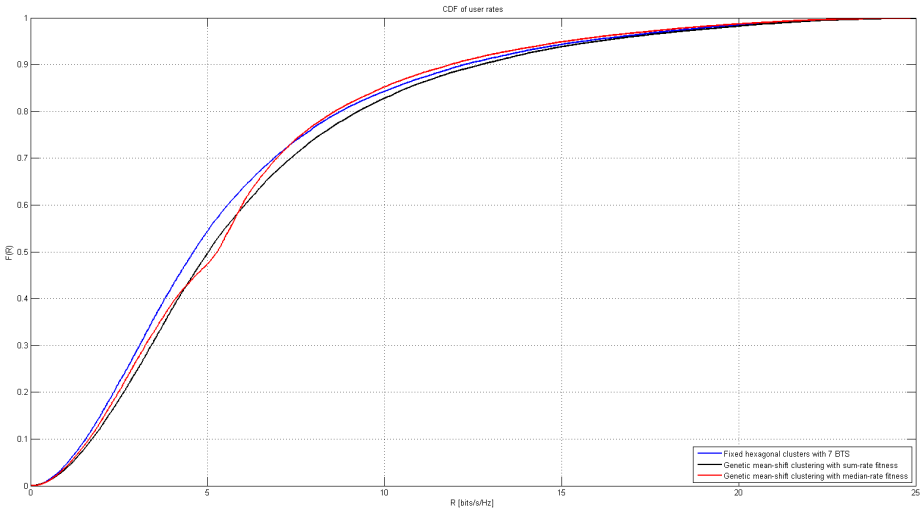
Figure 6.10: CDF of the user rates attained by evolutionary base-station clustering.

compare the results shown in the tables and bar charts obtained with the strict set of constraints with those obtained with the flexible one, we can see that the improvement we get by switching to the latter is pretty modest. As a consequence, this leads us to believe that, in order to obtain a real, significant improvement by tweaking the maximum cluster size constraints, we have to allow the average and median cluster sizes to increase significatively. In other words, we have to rise $L_{\max,a}$ and/or P in a more pronounced way.

Finally, apart from yet another graphical proof of the enormous performance gap existing



(a) Strict set of constraints.



(b) Flexible set of constraints

Figure 6.11: CDF of the user rates attained by genetic mean-shift clustering.

between the fixed scheme and our evolutionary clustering algorithms, a very interesting observation can be extracted from the shape of the CDF curves. If we compare the curves attained for median-rate fitness which those resulting from optimizing the average-rate, we can see that the CDFs corresponding to median-rate optimization have been distorted with respect to the typical shapes precisely in such a way that the median of the distribution is enhanced. Such distortion is also the reason why average-rate maximization was more balanced, increasing the median and sum rates in the same proportion; whereas median-rate maximization barely rised

the average-rate but enhanced the median-rate a lot. This is a fundamental consideration which shows empirically the incredible influence that the choice of one partition or another has in the overall system: we do not only can increase the mean or median user rates but, with a proper design, we have the power to change the whole PDF of the user rates at out whim. The flexibility we can attain by modifying the clusters seems endless.

As a side consideration, that effect also shows the rather impressive behavior of genetic algorithms: they have been able to converge to an anomalous region of the search space which dramatically increases the median rate on their own, just by imitating natural evolution.

Channel matrix based approaches vs location based approaches

Yet another fundamental observation is that, focusing now on the non-evolutionary algorithms, it seems that using the BTS and user locations to represent the signal propagation conditions outperforms the alternative of employing the channel matrix $\mathbf{H}$. In this way, we can see that mean-shift base-station clustering is on the lead followed closely by spectral base-station clustering with BTS-User distance similarity. This may seem a bit surprising since the channel matrix $\mathbf{H}$ contains more information after all. However, we already hinted at the reason why that occurs when we introduced the different similarity metrics for base-station clustering.

Imagine the following situation. Let us assume the existence of a toy scenario formed by just 3 cells, $\{b_1, b_2, b_3\}$. Furthermore, suppose that we want to obtain a partition with a cluster which contains 2 cells and another cluster containing just one. If we do it based on the channel matrix $\mathbf{H}$, we would end up grouping the two cells for which $\|(\mathbf{H})_{i,j}\|_F^2 + \|(\mathbf{H})_{j,i}\|_F^2$ is maximum. At first, we thought that this was a good approximation and it actually kind of worked, just with a performance poorer than expected.

To explain why our initial idea was not as good as we thought, we will continue with the previous example. Let us suppose now that we end up forming a cluster $\mathbb{S}_1$ with the pair of cells $\{b_1, b_2\}$ while leaving cell b_3 alone in another cluster $\mathbb{S}_2$. If we did that, it was because the mutual interference between cells b_1 and b_2 as measured by $\mathbf{H}$ was bigger than that between cells b_1 and b_3 or that between cells b_2 and b_3. However, it may perfectly occur that after computing the precoder $\mathbf{W_{tx}}$ and filter $\mathbf{W_{rx}}$ using that partition, the equivalent channel matrix $\widetilde{\mathbf{H}}$ contains a huge resulting interference between the clusters $\mathbb{S}_1$ and $\mathbb{S}_2$. Ironically, it may also happen that if we had formed, for instance, the cluster $\mathbb{S}_1$ with cells $\{b_1, b_3\}$ and cluster $\mathbb{S}_2$ with b_2, the resulting interference between clusters $\mathbb{S}_1$ and $\mathbb{S}_2$ was smaller than with the previous, seemingly optimal partition. In short, the introduction of $\mathbf{W_{tx}}$ and $\mathbf{W_{rx}}$ may transform $\widetilde{\mathbf{H}} = \mathbf{W_{rx}}\mathbf{H}\mathbf{W_{tx}}$ in something which does not even barely resembles $\mathbf{H}$. We already forecast that difficulty when we introduced this similarity function yet, to be honest, the problem has resulted to be much trickier than we had initially expected.

In conclusion, making any guess on the resulting shape of $\widetilde{\mathbf{H}}$ based on $\mathbf{H}$ alone is, at the very least, a risky approach. Because what may seem an optimal partition when looking at $\mathbf{H}$ may end up being a poor choice after obtaining the corresponding equivalent channel $\widetilde{\mathbf{H}}$,

those algorithms which were designed to work with $\mathbf{H}$ are outperformed by the ones which only employ the locations of base-stations and users within the scenario. While it is true that the latter actually have less information available to work with, they are much more robust and in the end that robustness weights more, at least according to our empirical results.

This constitutes yet another reason to show how complicated the problem at our hands is. We want to find the partition of the set of base-stations $\{b_i\}_{i=1}^{M}$ such that, for the particular signal propagation conditions at that instant, computing precoders $\mathbf{W_{tx}}$ and receive filters $\mathbf{W_{rx}}$ within each cluster in the aforementioned partition results in an optimal performance. However, the funny thing is that, as we have discussed just now in the previous example, the partition itself dramatically modifies the signal propagation conditions indirectly by changing the resulting set of precoders and receive filters.

This observation motivates the creation of a new line of research to develop similarity functions based on perturbation theory. Rather than measuring the total amount of interference, which is something that we cannot do if we do not know the partition yet, nor estimating it with $\mathbf{H}$, which is something which does not work too well, we could try to somehow base similarities on how much the total resulting interference would change if we reassigned a certain cell from one cluster to another. Nonetheless, this is a pretty complex problem since, in order for the resulting algorithm to be feasible, it should be purely analytic. That is, we cannot afford to reevaluate $\mathbf{W_{tx}}$ and $\mathbf{W_{rx}}$ hundreds of times. We would need to find a way of obtaining a closed form solution, or a reasonable approximation, to know how do $\mathbf{W_{tx}}$ and $\mathbf{W_{rx}}$ change when we introduce a small change in the partition. For now, we leave this idea as an open problem to be further developed in future projects.

As a side note, the reader has probably realized at this point that the normalized interference based similarity performs worse than any of the other three proposed similarity functions. The idea seemed sensible and we actually believe that it was worth trying. Nonetheless, our results confirm that it does not work so well as it is surpassed in performance even by the reference fixed scheme with hexagonal clusters of 7 cells.

Dynamic vs static tuning of parameters

According to the results shown before, the performance gain achieved when using a dynamic tuning of the algorithm's parameters is quite significant, even for those algorithms which would be actually feasible in a real system, such as basic mean-shift base-station clustering or spectral base-station clustering. Even if the margins obtained are not as spectacular as those attained by the evolutionary algorithms, they still are able to score close to a 10% median rate increase and 5% sum rate increase with respect to the fixed scheme with hexagonal clusters, which is something quite interesting if we take into account that the average and median cluster size actually decreases. Sadly, even if the algorithms are computationally feasible, the process required to dynamically tune the parameters is not. In order to do that, we would need to carry out a sweep on relatively big ranges for each of the parameters, which in the end amounts to

reexecuting the algorithm hundreds of times. As that is clearly something we cannot afford, we have used our results to study whether we can infer something about the optimal choice for the parameters of each algorithm. If we had a way to estimate such an optimal choice without relying on a brute-force search, our algorithms would become feasible and, thus, pretty interesting for their implementation in a real-life system.

As a reference, we will provide results related to the optimal parameters for mean-shift base-station clustering and spectral base-station clustering.

We will begin by discussing the mean-shift based algorithm. In figure 6.12 we show in the space (α, K), for each of the $N_{\text{rep}} = 100$ different scenarios, the particular values of α and K which optimized the performance measure which was being considered. As we can see, the good news we can extract from those plots is that the ranges we initially employed can be significatively reduced. Involving more than the 8 nearest neighbors in the calculation of the per-sample scalar bandwidths seems to be always suboptimal. Similarly, optimal values for α below 0.6 or above 1 are very rare. Sadly, the bad news is that within that reduced range, we are still clueless. Even if we have been able to reduce the complexity a bit, we need to carry out a parametric sweep within the new ranges as well and that remains an unfeasible approach.

On the other hand, in figure 6.13 we show the parameters of spectral base-station clustering in the space (α, β). Again for each of the $N_{\text{rep}} = 100$ different scenarios, what we see is which values of α and β optimized the sum-rate or the median-rate of the system. In this case, the situation is actually worse than before: we cannot even shrink the ranges. Virtually, any choice of parameters within the ranges we have considered could be the optimal and, indeed, even increasing the ranges a little bit seems like a good idea. Again, the complexity is far too much to be dealt with.

The bad side of all this is that finding a more efficient way to perform a dynamic tuning of the parameters is actually needed if we want our algorithms to be useful. Sadly, the results previously shown point out that when working with static tuning, none of our base-station clustering algorithms is worth being implemented in a real system. They do achieve a slight performance improvement while using average and median cluster sizes a bit smaller. Nevertheless, the gain is so modest that it does not make up for the pain of reconfiguring the whole network to introduce a clustering algorithm for dynamical cluster allocation.

Our base-station clustering algorithms are definitely really useful whenever they are optimally tuned for each particular scenario. Therefore, the need for figuring out more efficient ways to adjust the parametrization of our algorithms motivate yet another research line.

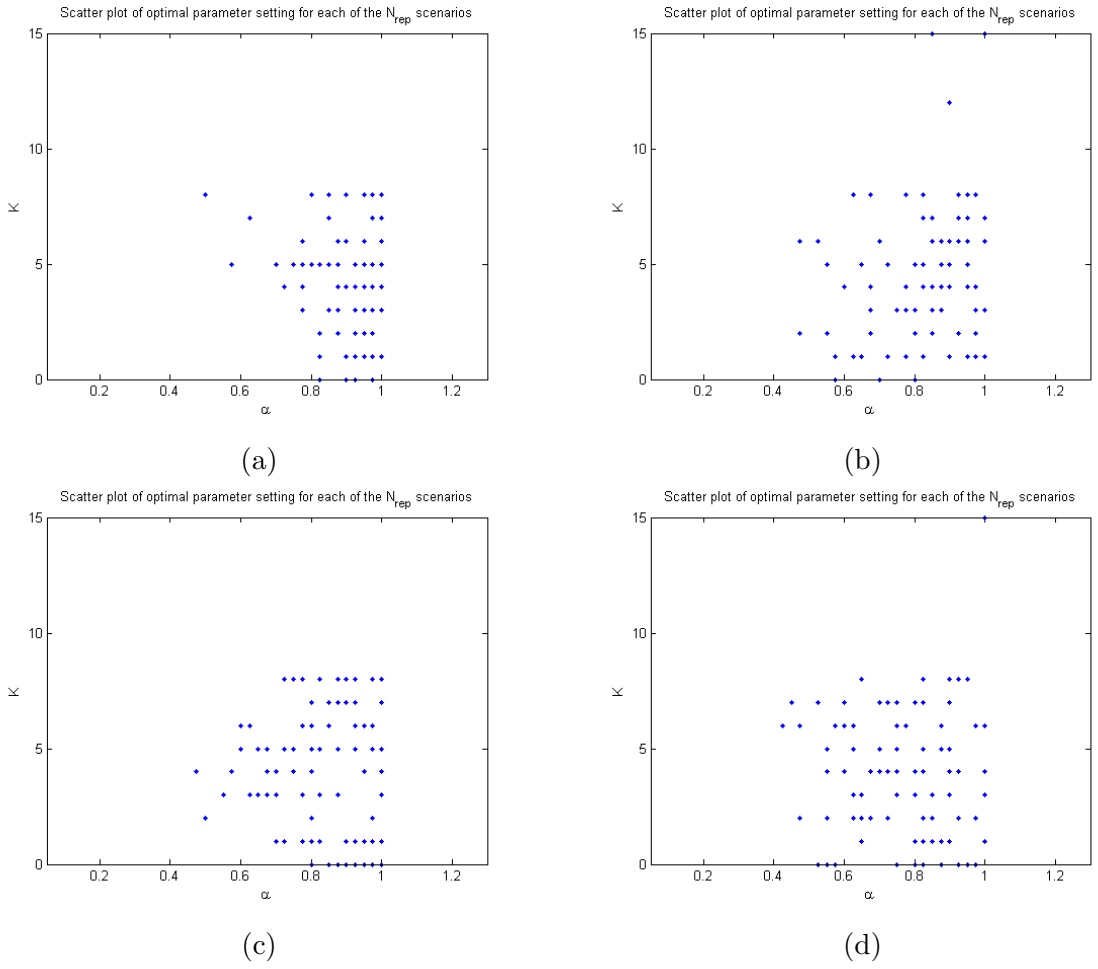


Figure 6.12: Scatter plot of the different 100 *resulting optimal parameters for basic mean-shift base-station clustering. In (a) we tuned for optimal sum-rate with flexible constraints. In (b) we tuned for optimal median-rate with flexible constraints. In (c) we tuned for optimal sum-rate with strict constraints. In (d) we tuned for optimal median-rate with strict constraints.*

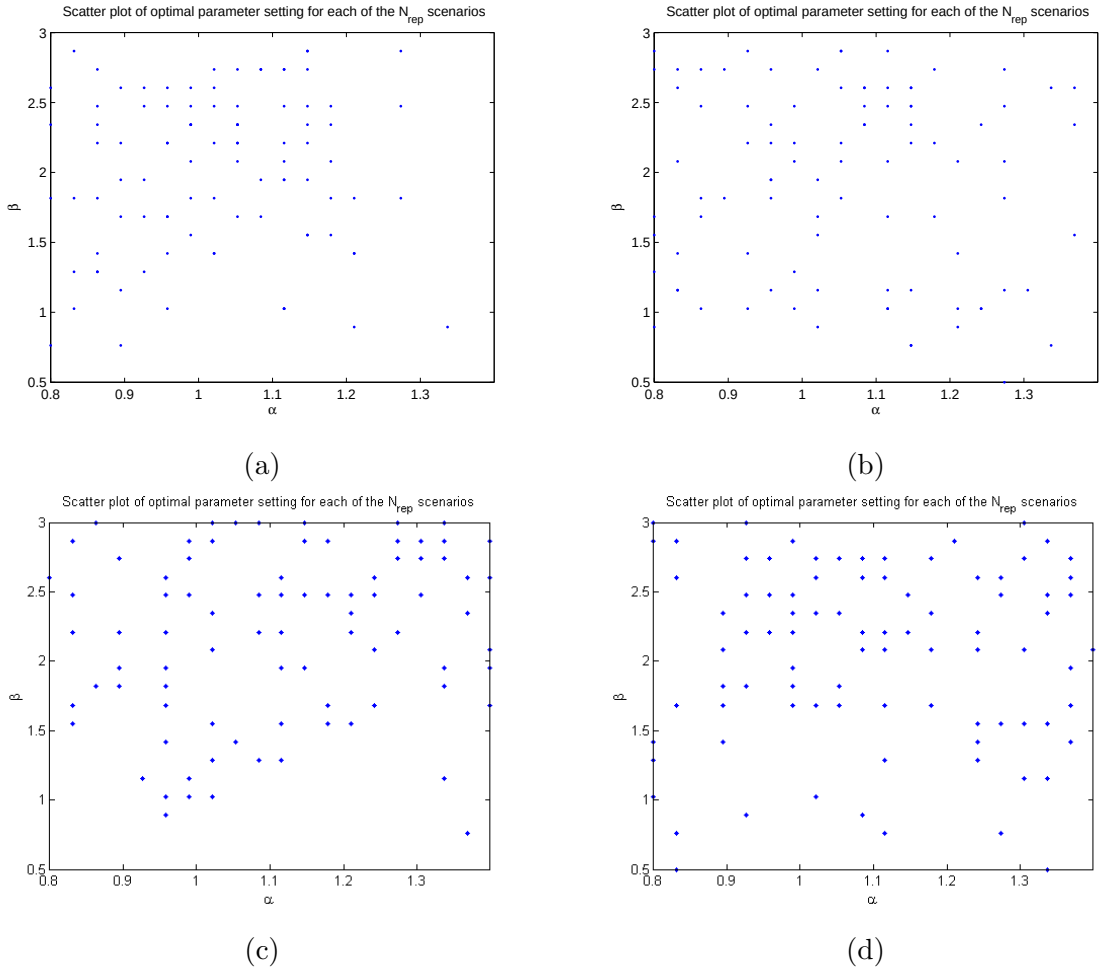


Figure 6.13: Scatter plot of the different 100 *resulting optimal parameters for spectral base-station clustering with BTS-User distance based similarity. In (a) we tuned for optimal sum-rate with flexible constraints. In (b) we tuned for optimal median-rate with flexible constraints. In (c) we tuned for optimal sum-rate with strict constraints. In (d) we tuned for optimal median-rate with strict constraints.*

Chapter 7

Conclusions and further studies

Within the field of MIMO linear precoding, what started as a mere review of the state-of-the-art for picking a scheme to be used in our simulations, turned out to be a throughout research process to develop new MMSE precoders for multi-cell coordinated MIMO.

On the one hand, we proposed a modification of the standard constrained optimization problem leading to MMSE solutions by including in the formulation not only the minimization of the intra-cluster mean-square error but, also, a reduction of the inter-cluster interference levels. We provided empirical evidence which proved that this modification leads to a dramatic increase of the system's throughput as well as to a considerable reduction of the total mean-square-error in distributed multi-user MIMO scenarios.

A review of the literature pointed out a lack of MMSE precoders apt for cellular coordinated MIMO. Apparently, the problem of minimizing the MSE with linear precoding subject to per-cell power constraints instead of the typical, well-known total power constraint for the whole system has not been solved yet. That was a strong source of motivation for trying to fill that gap ourselves, leading to the development of two different MMSE precoders with per-cell power constraints; one which simply extended the regular Wiener precoder and other which also introduced our interference-awareness idea. Nevertheless, even though most of the theoretical derivation has been completed, for now we are stuck at the very last step as the sought solution is hidden behind an implicit matrix equation which we have not been able to solve yet.

The results we achieved during our derivations were really interesting from a theoretical point of view as the structure of the implicit equations characterizing the analytical solution was very revealing of the inherent structure of multi-user distributed MIMO channels and their behavior. Not only that, everything seems to point out that if we were able to complete our work we would have found out a scheme surpassing the performance achieved by block diagonalization. Therefore, we believe that looking for the solution of the implicit matrix equation which determines the solution for our MMSE precoders, or even developing reasonable approximations, constitutes an open problem which is challenging but also very promising.

As we said, our primary goal has been researching whether the way in which cells are grouped in clusters has a noticeable impact in the resulting performance of the system. We have designed

several application-specific clustering algorithms to compare the results achieved by those with the performance of a system in which the clusters are always fixed and formed according to standard cluster shapes in cellular geometry, such as hexagons. The results achieved served as confirmation that the performance gap which can be potentially achieved by modifying the clusters is enormous.

Future lines of research

From a theoretical point of view, this project is a resounding success, as we have been able to prove that, all this time, we had in front of us a largely unexplored path which allows to radically enhance the performance of future cellular networks with a relatively moderate implementation cost. Nonetheless, neither any of the evolutionary algorithms nor the other algorithms under adaptive parametrization have a reasonable computational complexity. We believe that this provides a fairly strong motivation for opening new research lines destined to refine the work we began in this direction. The problem is certainly very complicated yet we also have several clear points where we can focus our effort.

On the one hand, we have developed a couple of evolutionary algorithms which, even if they are not feasible for a real system due to the slow convergence they exhibit, can provide us with the optimal partition given any scenario. In this way, we have a tool which allows us to create a huge database where each simulated scenario is stored along with its associated optimal partition. In this way, we have tools which can be used for trying to find out the relation between the signal propagation conditions and the optimal partition by reverse engineering.

Also, we have many algorithms, such as mean-shift base-station clustering, which are perfectly feasible yet we still need to find a way to dynamically tune its parameters without relying on a brute-force search in the parameter space. Our efforts to discover the principles governing the relation between the specific signal propagation conditions at each time instant and the optimal choice of parameters for any of our base-station clustering algorithms have been fruitless for now. Therefore, this remains a really interesting open problem to be tackled in future projects.

It has also been discussed that in cellular coordinated MIMO, the particular distribution of cells within clusters can vary the equivalent channel matrix much more than we had expected. As a consequence, methods which try to assign cells to clusters based on the original channel matrix do not achieve satisfactory results. It would be a very interesting research line trying to develop a perturbation analysis theory which was able to estimate how would the resulting equivalent channel matrix vary when moving a cell from one cluster into another. If we had that, we would open endless possibilities for developing similarity functions and clustering algorithms much better adapted to our context.

Most of the algorithmic and conceptual framework built along this project still holds when using directive antennas. However, as we discussed previously in this document, the parametrization of all the algorithms would increase in complexity and, in most cases full covariance matrices would be needed. Since real-world cellular systems usually employ some kind of sectorization,

studying this open problem seems to be a very practical research line.

Probably, the fundamental pillar of this project is the observation that in cellular systems with base-station coordination the main source of performance degradation comes from inter-cluster interference. Based on that perception, we have developed MIMO precoders which attempt to limit the interference levels they induce in neighboring clusters and, more importantly, we have carried out a throughout study of the impact of the cluster design on the overall performance of the system. Moreover, due to the effects of propagation loss, inter-cluster interference is really severe only on the cluster boundaries. Motivated by that observation, it would be really interesting if we had a way to eliminate boundaries at all: i.e. employ a soft-coordination scheme in which any given cell could belong to several clusters at the same time. Nevertheless, when approaching the problem from a clustering perspective, that merely amounts to switching from partitional clustering algorithms to overlapping clustering algorithms. During this project, we have tried to test our hypothesis and give the first steps on this research line, hence we decided to stick to simpler schemes which only work with hard-partitions. However, once the usefulness of carefully designing the clusters for cellular coordinated systems has been proven, changing the paradigm to a solution based on soft-partitions appears to be the best choice yet it would result in a considerable increase in the complexity of the problem. Therefore, an extension of this project towards non-exclusive clustering algorithms is a fascinating research line for a mid-term future.

Appendices

Appendix A

Interpretations of spectral clustering

In section 4.4, we developed spectral clustering using graph theory, while trying to solve graph cut problems. However, as an example, other formulations leading to spectral clustering are:

Relation to random walks

A random walk on a graph is a stochastic process which jumps in a random pattern from one vertex to another. The transition probability of jumping from vertex $\mathbf{v}_i$ to vertex $\mathbf{v}_j$ is given by $p_{ij} = w_{ij}/d_i$. Then, the $N \times N$ transition matrix $(\mathbf{P})^{i,j} = p_{ij}$ can be computed in a compact way as $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$. Note that the normalized graph Laplacian $\mathbf{L}_{rw}$ satisfies $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{P}$, so that $\mathbf{P}$ and $\mathbf{L}_{rw}$ share eigenvectors whereas their spectra satisfy $\sigma(\mathbf{L}_{rw}) = 1 - \sigma(\mathbf{P})$. This gives a first hint on the strong relation existing between spectral clustering and random walks in a graph.

In [50], the authors found a very nice link between NormalizedCut and random walks. From now on, we will focus on graphs which are connected and non-bipartite, since this ensures that the random walk has a unique stationary distribution $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)^T$ (do not confuse $\boldsymbol{\pi}$ in this section with the vectors of weights in PenalizedCut). It can be shown that, under the two restrictions imposed in the graph properties, we have $\pi_i = d_i/\mathrm{vol}(V)$. Then, we can prove the following theorem:

Theorem 6. *If G is a connected, non-bipartite graph and we run a random walk $X[n] \mid n \in \mathbb{N}$ starting with $X[0]$ in the stationary distribution $\boldsymbol{\pi}$, then for a subset V_i of V we have that:*

$$\frac{W\left(V_i, \overline{V_i}\right)}{\mathrm{vol}\left(V_i\right)} = P(\overline{V_i}|V_i)$$

Where we have defined $P(B|A) = P\left(X[1] \in B | X[0] \in A\right)$ for $A, B \subset V$.

Proof. By definition of conditional probability,

$$P(B|A) = \frac{P\left(X[1] \in B, X[0] \in A\right)}{P\left(X[0] \in A\right)}$$

On the other hand, we have that:

$$P\left(X[1]\in B, X[0]\in A\right)=\sum_{i\in A}\sum_{j\in B}P\left(X[0]=1,X[1]=j\right)=\sum_{i\in A}\sum_{j\in B}\pi_i p_{ij})=$$
$$=\sum_{i\in A}\sum_{j\in B}\frac{d_i}{\mathrm{vol}(V)}\frac{w_{ij}}{d_i}=\frac{1}{\mathrm{vol}(V)}\sum_{i\in A}\sum_{j\in B}w_{ij}$$

Similarly, $P\left(X[0]\in A\right)=\mathrm{vol}\left(A\right)/\mathrm{vol}\left(V\right)$. Putting both things together, and recalling the definition:

$$W\left(A,B\right)=\frac{1}{2}\sum_{\{i|\mathbf{v}_i\in A\}}\sum_{\{j|\mathbf{v}_j\in B\}}w_{ij}$$

We conclude that:

$$P\left(X[1]\in B|X[0]\in A\right)=\frac{W\left(A,B\right)}{\mathrm{vol}\left(A\right)}$$

□

Since $W\left(V_i,\overline{V_i}\right)/\mathrm{vol}\left(V_i\right)$ is the factor in NormalizedCut associated to cluster V_i, we have the following interpretation of normalized spectral clustering: Normalized cut tries to find a partition such that a random walk in the associated graph jumps very rarely between distinct clusters.

Minimum-variance criteria

Another reasonable approach to clustering is the so called minimum-variance clustering. The idea is that we seek a partition such that the variance of each cluster is as small as possible. This is equivalent to requiring tight clusters, so that the similarity between points in the same cluster is high. A mathematical way of expressing this is through the minimization of the trace of the within-class covariance matrix:

$$\mathbf{S}_W=\frac{1}{N}\sum_{j=1}^{K}\sum_{\{i|\mathbf{x}_i\in\mathbb{S}_j\}}\left(\mathbf{x}_i-\boldsymbol{\mu}_j\right)\left(\mathbf{x}_i-\boldsymbol{\mu}_j\right)^T \tag{A.1}$$

Where $\boldsymbol{\mu}_j$ is the center of mass of the j-th cluster as defined in equation (4.12).

The clustering algorithms which obtain partitions trying to minimize $\mathrm{Tr}\left(\mathbf{S}_W\right)$ are called minimum-variance methods.

Interestingly enough, in [3], the authors discovered a very strong connection between spectral clustering and minimum-variance clustering. Indeed, by defining a weighted within-class covariance matrix in a RKHS induced by a reproducing Mercer kernel K.

A Mercer kernel is a mapping $K:\mathbb{X}\times\mathbb{X}\mapsto\mathbb{R}$ such that $K\left(\mathbf{x}_i,\mathbf{x}_j\right)=\langle\boldsymbol{\psi}(\mathbf{x}_i),\boldsymbol{\psi}(\mathbf{x}_j)\rangle$ for some unknown mapping $\boldsymbol{\psi}:\mathbb{X}\mapsto\mathcal{H}$, from the original data set into the RKHS. The fact that

the mapping $\boldsymbol{\psi}$ does not need to be known is usually referred to as the "kernel trick". The corresponding points in the RKHS, $\tilde{\mathbf{x}}_i = \boldsymbol{\psi}(\mathbf{x}_i)$ are usually called *feature vectors* and define a new data set $\tilde{\mathbb{X}} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_N\}$, characterized by the data matrix $\widetilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1 \tilde{\mathbf{x}}_2 \ldots \tilde{\mathbf{x}}_N]^T$

The idea know is that, by using the kernel "trick", we can obtain the within-class covariance matrix in $\mathcal{H}$, $\widetilde{\mathbf{S}}_W$, without explicitly knowing the mapping $\boldsymbol{\psi}$ nor the data matrix in the RKHS $\widetilde{\mathbf{X}}$. Indeed, (see [3]):

$$\mathrm{Tr}\left(\widetilde{\mathbf{S}}_W\right) = \mathrm{Tr}\left(\mathbf{E}^T\boldsymbol{\Pi}\mathbf{H}_\pi^T\boldsymbol{\Delta}\mathbf{H}_\pi\boldsymbol{\Pi}\mathbf{E}\left(\mathbf{E}^T\boldsymbol{\Pi}\mathbf{E}\right)^{-1}\right) \tag{A.2}$$

Where:

$$\mathbf{H}_\pi = \mathbf{I}_N - \frac{1}{\boldsymbol{\pi}^T\mathbf{1}_N}\boldsymbol{\pi}\mathbf{1}_N^T \tag{A.3}$$

And:

$$(\boldsymbol{\Delta})^{i,j} = K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j) \tag{A.4}$$

Doing some algebraic manipulations, the authors in [3] were able to prove the following theorem, which we shall enunciate here without proof:

Theorem 7. *Minimum-variance clustering in a RKHS induced by a Mercer kernel K, which generates the kernel matrix $(\mathbf{K})^{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$, understood as the minimization of $\mathrm{Tr}\left(\widetilde{\mathbf{S}}_W\right)$ where each data sample $\mathbf{x}_i$ is weighted according to a custom weight π_i, is fully equivalent to solving a PenalizedCut problem with weights π_i characterized by the "unnormalized graph Laplacian" $\mathbf{L}$ satisfying:*

(a) $\mathbf{L}^+ = \mathbf{K}$, *where* $\mathbf{L}^+$ *denotes the Moore-Penrose pseudoinverse.*

(b) $\mathrm{rank}(\mathbf{L}) = \mathrm{rank}(\mathbf{K}) = N - 1$, *with N being the number of points in the data set.*

From theorem 7 we obtain a fundamental intuition about the operation of spectral clustering, since it shows that there exists a Mercer Kernel such that the solution of the minimum-variance criterion in the corresponding RKHS equals penalized cut. As a technical note, when we use the "reverse" interpretation, obtaining a PCut problem which equals a minimum-variance problem, we must take into account that the matrix $\mathbf{L}$ obtained from $\mathbf{K}$ may not fulfill the properties of a graph Laplacian. However, the theorem shows the duality between minimum-variance criterion and the "fake" graph induced by $\mathbf{K}$ still holds.

Margin-based criteria

For the remainder of this section, we assume that the reader has some basic understanding on the operation of Support Vector Machines (SVM). If that is not the case yet they are interested in following this section, a very gentle introduction to the fascinating topic of support vector machines can be found in [51].

Apart from the minimum-variance view, the authors in [3] were able to find one of the most insightful interpretations of spectral clustering up to now. They showed that it is possible to think of spectral clustering in terms of finding hyperplanes which maximize margins in a RKHS, coming up with a beautiful link between spectral clustering and support vector machines. According to their results, spectral clustering with non-maximum suppresion rounding uses the data set to find a set of $k-1$ hyperplanes which separate the data in some RKHS which maximal margin. However, since there are no labels available, the optimization problem to find the hyperplanes must be simplified, dropping the sum term which penalizes incorrect classifications.

In particular, they showed that the same relaxed problem which appears when solving both PCut and the minimum-variance criterion is also equivalent to:

$$\min_{\mathbf{S}} \ \mathrm{Tr}\left(\mathbf{S}^T\mathbf{S}\right) \quad s.t. \quad \mathbf{S}^T\widetilde{\mathbf{X}}^T\mathbf{\Pi}\mathbf{1}_N = \mathbf{0}, \quad \mathbf{S}^T\widetilde{\mathbf{X}}^T\mathbf{\Pi}\widetilde{\mathbf{X}}\mathbf{S} = \mathbf{I}_{k-1} \tag{A.5}$$

With $\mathbf{S} = [\mathbf{s}_1\mathbf{s}_2\ldots\mathbf{s}_{k-1}]$ being a matrix whose columns are the hyperplanes in the RKHS and $\widetilde{\mathbf{X}}$ is the data matrix of feature vectors, following the convention of arranging observations by rows and attributes by columns. Note that the minimization of $\mathrm{Tr}\left(\mathbf{S}^T\mathbf{S}\right)$ is fully equivalent to the minimization of the unweighted sum of the norms of the hyperplanes, exactly as it is done when formulating the constrained optimization problem which leads to support vector machines. Moreover, the condition $\mathbf{S}^T\widetilde{\mathbf{X}}^T\mathbf{\Pi}\mathbf{1}_N = \mathbf{0}$ requires that the hyperplanes pass through the weighted centroid of the data $\sum_{i=1}^N \pi_i\tilde{\mathbf{x}}_i$.

At this point, we should point out that this discussion only holds when we use the non-redundant PCut formulation of [3], since we are assuming $k-1$ hyperplanes only. However, this derivation could also be extended to the redundant case.

One of the many interesting properties of support vector machines is that resulting maximal-margin hyperplanes are linear combinations of the feature vectors, $\mathbf{s}_i = \sum_{j=1}^N \beta_{ji}\tilde{\mathbf{x}}_j$. This can be written in matrix form as $\mathbf{S} = \widetilde{\mathbf{X}}^T\mathbf{B}$, with $(\mathbf{B})^{i,j} = \beta_{ij}$ a $N \times (k-1)$ matrix of regression coefficients. It is possible to show (see [3] for details), that in order for this problem to be equivalent to the PenalizedCut, we must do the change of variables $\mathbf{H} = \mathbf{K}\mathbf{B}$ where $\mathbf{K}$ is the kernel matrix and $\mathbf{H}$ is the solution of the *relaxed* PCut problem. Since $\mathbf{K} = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T$, we have that $\mathbf{H} = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T\mathbf{B} = \widetilde{\mathbf{X}}\mathbf{S}$. But then, this means that $(\mathbf{H})^{i,j} = \mathbf{s}_j^T\tilde{\mathbf{x}}_i$.

That is a fundamental results, which establishes that the entries of the solution $\mathbf{H}$ of the *relaxed* PCut problem (and, according to the previous section, *relaxed* minimum-variance clustering problem), can be understood as signed distances between the data samples in the RKHS $\tilde{\mathbf{x}}_i$ and the hyperplanes $\mathbf{s}_j$.

With this idea, spectral clustering with non-maximum suppression rounding can be thought to be a two-step process in which we first fit a set of $k-1$ hyperplanes in the data following an optimization problem similar to that of support vector machines (but unsupervised) and later, we "classify" the samples by assigning them to the hyperplane which gives a bigger margin. A dummy example can be seen for an artificial three-class problem in figure A.1, where we can see the two hyperplanes ($k = 3$) found by spectral clustering and the corresponding samples in the

feature space. The reader can observe how they effectively separate the three clusters.

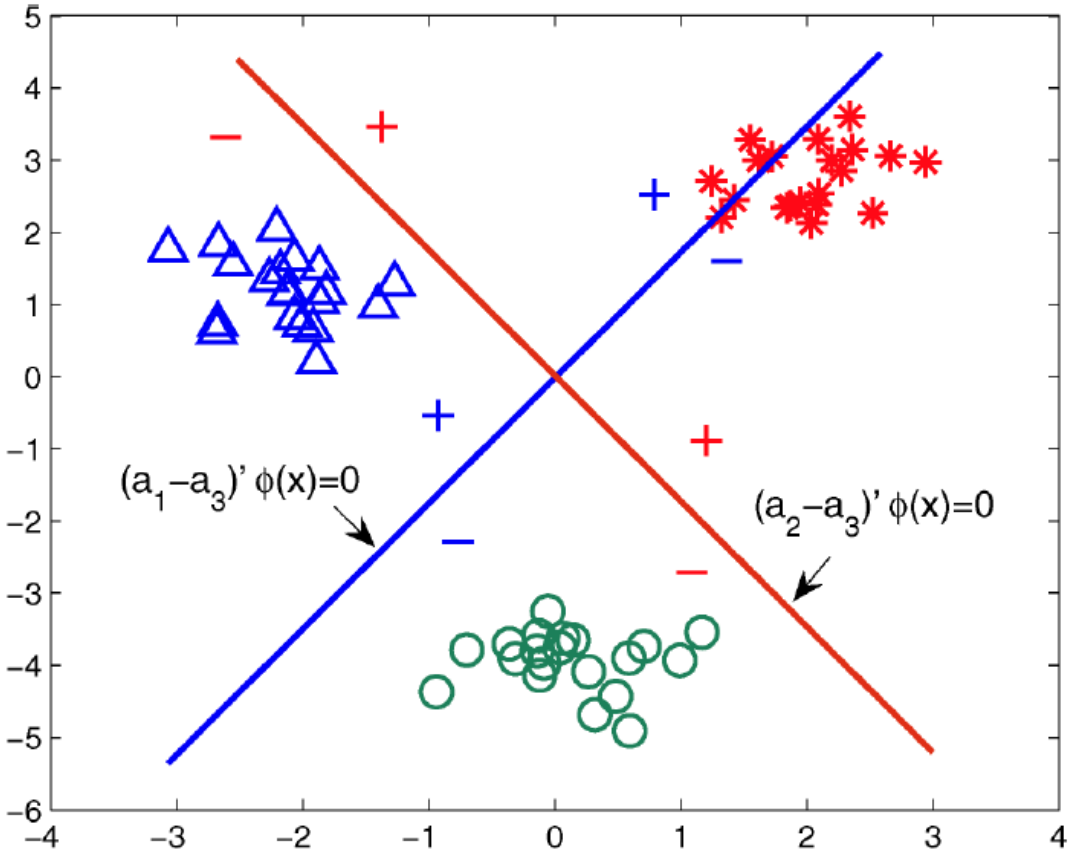


Figure A.1: Illustration of the margin-based perspective of spectral clustering for a three-class separable artificial data set. Image taken from [3].

Appendix B

Biological background for genetic algorithms

During this section, we aim to provide an intuitive, very non-rigorous discussion on the biological processes which motivated the creation of genetic algorithms, so that people with an interest in machine learning can get a grasp on the basic concepts and terminology used in this when dealing with this family of optimization algorithms. By no means we intend to provide here a text which has value for gene researchers. Most of this introduction is based on two divulgative tutorials found online: [52] and [53]. Readers willing to know more about this fascinating topic can read selected chapters of [54], taking into account that a certain background on biochemistry is required. In a fast reading, this section can be skipped since it is not fundamentally related to the purpose of this project but rather, attempts to provide a complete description of the algorithms used.

Genetic algorithms are, obviously, somehow related to *genes*. Roughly speaking, a *gene* is a set of instructions for building a specific protein.

Proteins are a family of large, complex molecules which are fundamental to all the working and structural aspects of a living being. For example, insulin, which regulates blood sugar; melanin, responsible for hair's and skin's color; antibodies, responsible for protecting the organism against viruses and bacteria; or collagen, which forms the structural scaffolding of many tissues, are proteins. Other proteins regulate the body's production of proteins or carry out other regulatory functions. For instance, messenger proteins coordinate biological processes between different cells, tissues, and organs by transmitting signals and enzymes carry out most of chemical reactions that take place in cells. Without proteins, living things (at least those we have had the opportunity to study) simply could not exist.

Therefore, genes usually affect an organism's traits indirectly. There may not be a gene which says that human beings shall have hands with five fingers. However, within the whole human genome, there exists a set of genes, maybe distributed in many different chromosomes, which will control the existence of hands and the number of fingers by telling the cells in the

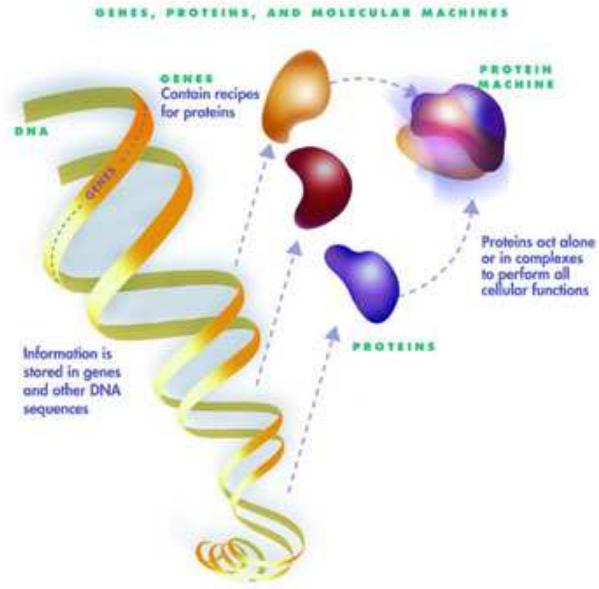


Figure B.1: Genes contain instructions for building the molecules which control working and structural aspects of a life form: proteins.

body which proteins have to be made, how to build them, which amount, when and where.

Genes are made of a huge and very complicated molecule called DeoxyriboNucleic Acid (DNA). DNA itself is built from smaller molecules denoted nucleotides. Those have three parts: a phosphate molecule, a sugar molecule and a nitrogenous base. There are four possible bases in the DNA: adenine (A), thymine (T), guanine (G) and cytosine (C).

The DNA molecule exhibits the very famous double helix shape. It can be seen as a ladder whose sides have been twisted into a spiral, as in figure B.2. Those sides of the ladder are made of phosphate and sugar, in such a way that the sugar of one nucleotide is linked to the phosphate of the next. On the other hand, the rungs of the ladder are made of two nitrogenous bases linked together in the middle, as shown in figure B.3.

The nitrogenous bases forming the rungs are the ones that carry the genetic information. More precisely, the ordered sequence of bases is the one encoding biological data. However, there is a restriction in the way the bases are paired: cytosine can only be linked with guanine (C-G) and thymine with adenine (T-A). Any other combination would result in an unstable molecule. Since the order of the bases on one side of the ladder determine the order on the other side, DNA molecules contain a duplicate of the genetic information. Therefore, when expressing a DNA sequence, it is written as if it was single-stranded, that is, only the sequence of nitrogenous bases in one side of the ladder is written. For instance, if we write TGTCGGATTAACGCCTT, then we know that ACAGCCTAATTGCGGAA would be the corresponding sequence in the other side.

Nonetheless, such a redundancy is by no means useless. The fact that each DNA strand con-

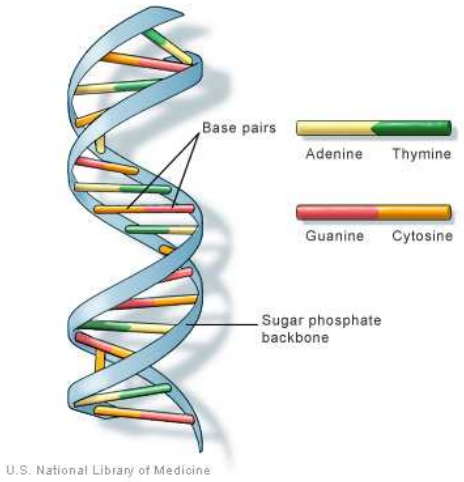


Figure B.2: DNA molecule has the shape of double helix formed by nitrogenous base pairs attached to a sugar-phosphate backbone.

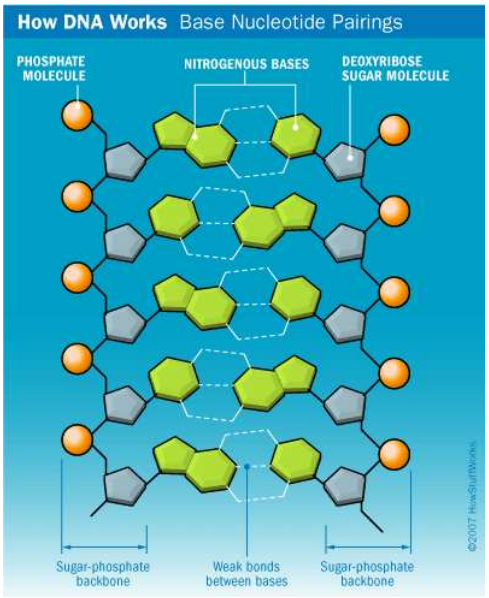


Figure B.3: Detail of how nucleotides are linked to form a DNA molecule.

tains the information necessary to reconstruct the other, gives raise to one of the most important properties of this molecule: it can create copies of itself. Whenever a cell needs to create a copy of a part or all its DNA, it splits the molecules, breaking the rungs of the ladder apart by the weak bonds between the bases. While this process occurs, there are lots of nucleotides, happily floating free in the cell. When the two strands are separated, those nucleotides can attach to complementary nucleotides in each of the strands, creating, if everything goes right, two exact copies of the original molecule.

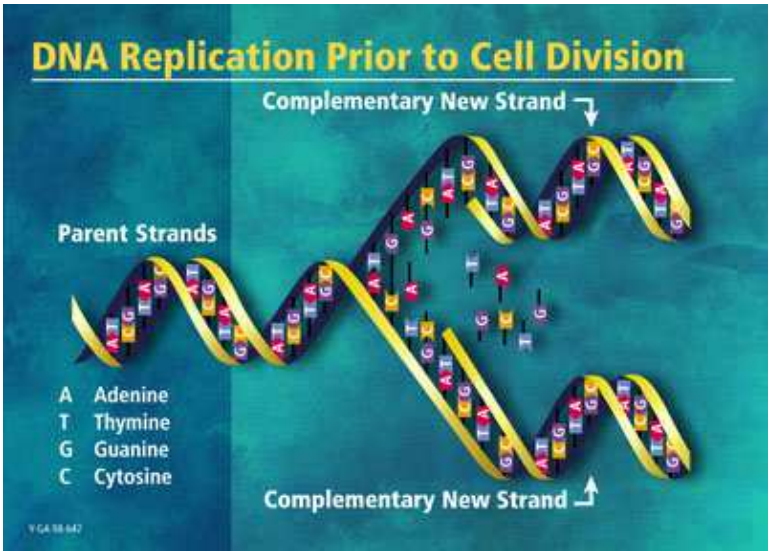


Figure B.4: The DNA molecule can replicate itself by separating both strands and reconstructing each copy with complementary nucleotides.

As we said, DNA essentially tells cells how to make proteins. Proteins are made up of sequences of *amino acids*. DNA sequences can be "translated" from the nucleotide language of DNA into the amino acid language of proteins. There exist 20 different amino acids. The genetic code uses groups of three bases, referred to as *codons*, to encode a particular amino acid. Since DNA uses a four-symbol alphabet, with a vector of three elements we can potentially encode up to $4^3 = 64$ amino acids. This makes the genetic code even more redundant.

Not all the genes, nor all the DNA in a gene, are actually a recipe for a particular protein. A gene actually has several parts. The protein-making instructions are split into short sections called *exons*, which are interspersed with longer sections of "extra" or "junk" DNA, denoted *introns*, whose function is not fully understood yet. Despite what their name suggests, recent investigations point out that "junk" DNA may be fundamental for a complete understanding of genetics, hence the original denomination could end up being a rather unfortunate choice. More importantly, genes also contain regulatory sequences, which help determine where, when, and in what amount proteins are made. All the cells in a living being contain the whole genome. Nonetheless, different cells perform different functions because each type of cells only fabricates a subset of the proteins for which they have the "blueprints". Regulatory sequences specify

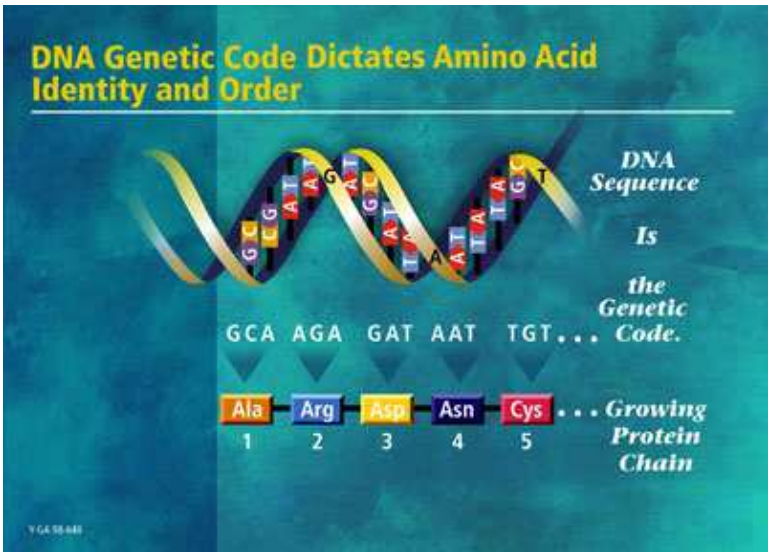


Figure B.5: Codons in a DNA sequence code for amino acids to build a specific protein.

which genes are "switched on", transmitting their protein-making instructions to the rest of the cell, and which are "switched off", hence allowing for cell specialization in complex organisms such as human beings.

The set of genetic material in a living thing is usually "organized". The genes are packed into thread-like structures known as *chromosomes*. The support of the chromosomal structure is made up from proteins called histones, which contain the DNA forming the genes tightly coiled many times around it, as shown in figure B.6.

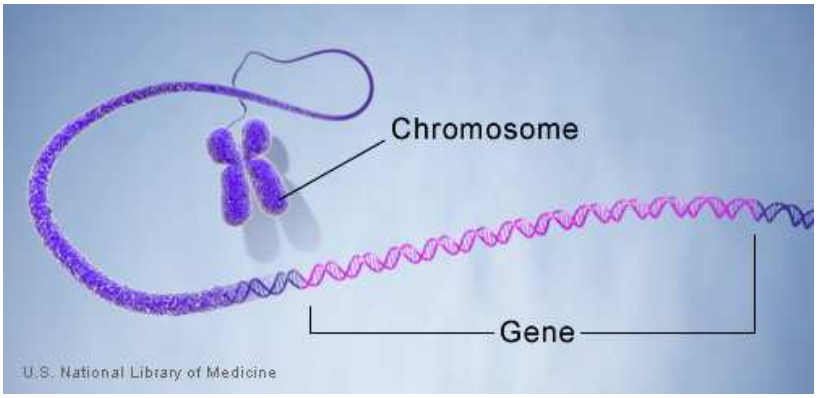


Figure B.6: Genes are made up of DNA. Each chromosome contains many genes.

Different organisms have a different number of chromosomes. Simple life forms, such as bacteria, may have a single chromosome containing all the genetic material. However, more complex living things, like mammals, usually have several. For instance, human beings have 46. Chromosomes are usually shown in charts called *karyotype*, which contain photographs of stained chromosomes in the organism arranged in order of size.

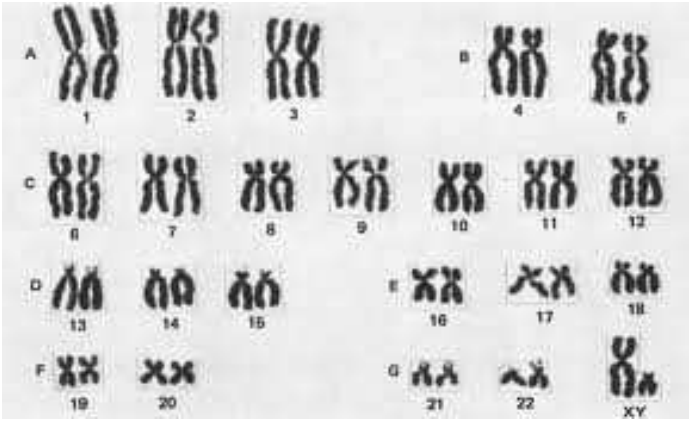


Figure B.7: A karyotype for a normal human being.

Moreover, in many organisms, chromosomes come in pairs, in such a way that both members of a pair have the same shape and size, and exhibit the same banding patterns when stained with fluorescent dyes. Such organisms are called *diploid*. As an example, human beings are diploid organisms with 23 pairs of chromosomes. Organisms with unpaired chromosomes are said to be *haploid*. When two chromosomes are members of the same pair, they are said to be *homologous chromosomes*.

The genes packed within the chromosomes determine what an organism is and what are its traits. But, how does a new cell obtain the genes which regulate its behavior?

New cells are created through cell division, which can be of two types: mitosis and meiosis.

The term "cell division" usually refers to mitosis, which is the process by which new cells are created within an already existing organism. During the process of mitosis, a cell creates a duplicate of all its contents, including the chromosomes, and then splits into two exact copies of the mother cell. The duplication of the genetic material is possible thanks to the redundancy of the DNA molecule, which allows to reconstruct one copy from each strand.

The mitosis process is critical and is throughly controlled by a few genes. Usually, everything goes as expected and the replication process is successful. However, random errors in the DNA replication process can occur: that is, a mutation can happen. Most mutations are small-scale mutations. For instance, in point mutations, a single nucleotide in the chain gets exchanged by another; in insertion, a few extra nucleotides are added to the DNA sequence; and in deletion, some nucleotides are erased from the sequence. Large-scale mutations can also occur, like duplication of some chromosomal regions, chromosomal inversions, by which the order of the DNA sequence in a big portion of DNA gets reverted or chromosomal translocations, where two different chromosomes exchange their genetic material. All those mutations contribute to changing the original genetic material, which also changes the behavior and functionality of the cell. In many occasions, if the changes are small, the effects can be negligible. In other cases, the consequences of failure in the DNA replication process may be catastrophic for the organism, for instance, certain cancers are induced through mutation. Nonetheless, it may also happen

that the change is positive and gives the mutated cell a trait which enhances its functionality. Mutations are random, and so are their effects.

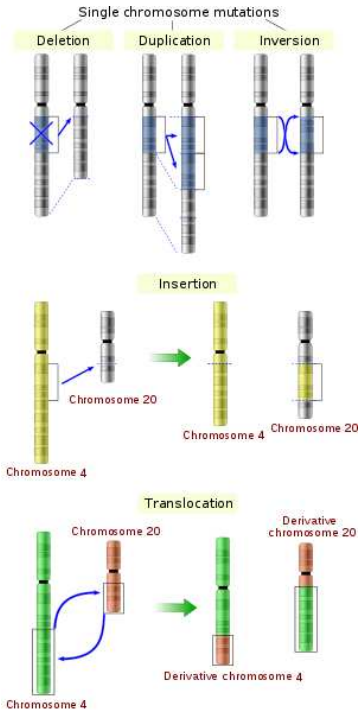


Figure B.8: Some examples of mutations.

On the other hand, when new organisms are created, they obtain their chromosomes from their parents. For living things that reproduce asexually, such as bacteria, reproduction is as simple as dividing in two. The genetic material is passed onto the next generation through cell division.

However, organisms which reproduce sexually have a more complex genetic inheritance process, which is where meiosis, the other type of cell division, shows up. Meiosis is essentially the formation of sex cells, sperm and egg. It is a two-step process in which four sex cells are created, each containing half the normal number of chromosomes in the organism. For instance, human sex cells contain only 23 chromosomes. When sperm and egg cells unite during conception to create a new individual, the normal number of chromosomes is restored, with each parent cell contributing half the chromosomes. This explains why chromosomes usually come in pairs: one homologous chromosome comes from the mother whereas the other one comes from the father. That is also the reason why, in sexual reproduction, children exhibit a mixture of the biological traits of mother and father.

The first step in meiosis, denoted as meiosis I, is very similar to mitosis. The DNA first replicates, creating one copy of each chromosome, so that the cell splits in half creating the two daughter cells. However, the fundamental difference with respect to mitosis is that the two new cells are not a perfect duplicate of the mother cell. This is because, during meiosis, *crossing-*

over occurs. Crossing-over is essentially an exchange of genes between each pair of homologous chromosomes. In other words, during mitosis, for each chromosome pair, the cell creates one exact copy of the chromosome inherited from the father, and another copy of the chromosome inherited from the mother. However, during meiosis, the maternal chromosome gets mixed with the paternal chromosome before replication, so that the two daughter cells contain genetic material which is different than that from both its sibling and mother.

In meiosis II, the second step, each of the two daughter cells are subdivided without any DNA replication process, so that four sex cells with only half the number of chromosomes are created. Therefore, each of the four sex cells which can lead to the conception of a new organism actually contains a different, random mixture of its grandparent's genetic material. Moreover, as in mitosis, random errors leading to mutation can occur at any step, leading to the creation of new traits not present in the lineage.

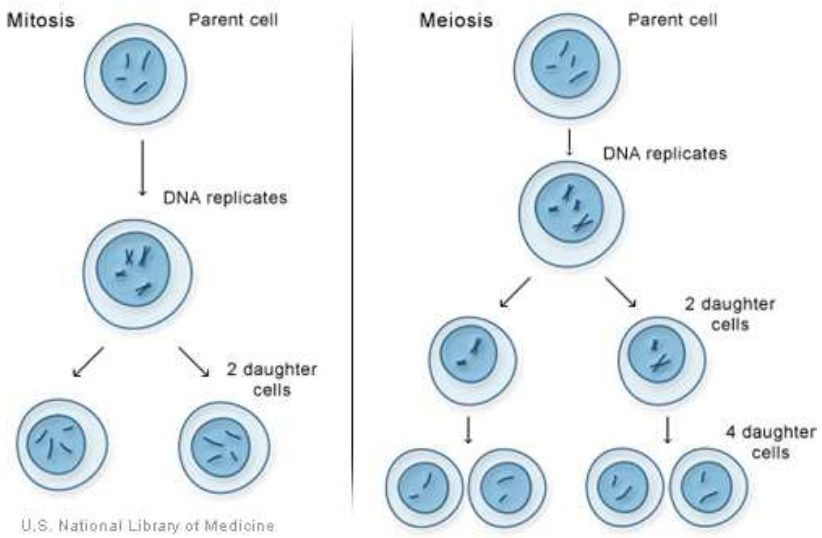


Figure B.9: Mitosis vs meiosis.

Crossing over during meiosis allows children of the same parents to have different traits. This dramatically increases genetic variability, which is the main reason why all complex life forms use some kind of sexual reproduction.

Reproduction ensures that individuals tend to be like their parents, that is, traits are inherited. However, mutations and crossing over allow the genetic code of each individual to be slightly different, introducing variation. In most cases, those changes will be irrelevant, or even negative. However, from time to time, a variation in the genome of an individual may grant a new trait which enhances the organism's survivability against the average individual of its species. Individuals whose genome makes them weaker, will have a harder time to find a mate to reproduce and pass its poor genome onto the next generation. Actually, the weaker they are, the harder it gets, being the extreme case those in which they are prone to disease and die

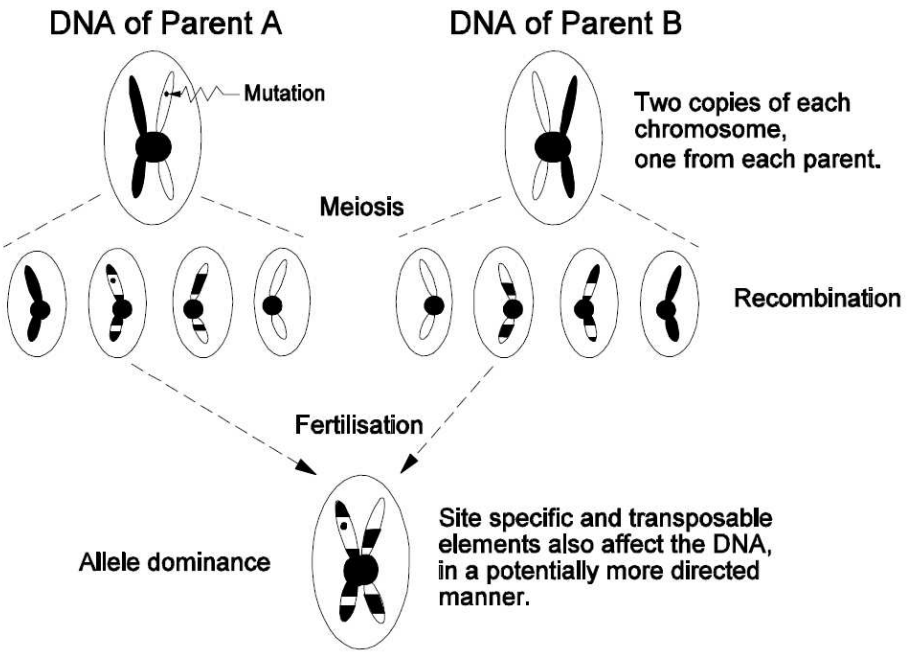


Figure B.10: Sexual reproduction through meiosis.

young. On the contrary, the stronger individuals will survive longer and have lots of children.

Charles Darwin was the first one to realize that, if this combination of inheritance of traits with some variability and the survival of the fittest process was iterated during a really long time, after countless generations, extremely complex organisms could be originated from very simple living things such as bacteria.

From the point of view of engineering, life can be seen as a game in which we have to find the optimal genome which codes for the best possible organisms. If we think about it that way, we realize that natural evolution is the most fascinating and powerful optimization algorithm that is and, probably, will be. It has been able to create human beings, extremely complicated organisms which are close to achieving the knowledge to artificially manipulate life, out of something as trivial as unicellular organisms.

It was only a matter of time until someone thought that, if natural evolution worked so well in biology, it could be also applied to computation. And so, evolutionary computing was born.

Evolutionary computing mimics natural evolution incredibly well, simultaneously being the source of their greatest strength, and their greatest weakness. They are really powerful optimization algorithms able to find a nice solution to any optimization problem without any a priori knowledge, but they are as painfully slow as their biological counterparts. Natural evolution took billions of generations of new individuals to create human beings. However, in practically all real-world applications, we cannot afford to simulate so many iterations of the algorithm until we achieve the desired result. This usually rules out the usage of genetic algorithms except for a few scenarios.

Appendix C

Performance of interference aware Tx-WF vs standard Tx-WF

In this appendix, we will try to provide the reader with some results to illustrate the usefulness of the concept of Interference Aware filters we have developed during this project. As the reader may recall, during chapter 3, several new MIMO precoding schemes were introduced.

First of all, we discussed the Generalized Tx-ZF, a slightly modified version of the regular Tx-ZF based on inverting the channel with the Moore-Penrose pseudo-inverse. In this new scheme we proposed, an implicit form of per stream power allocation was included in the optimization problem to maximize the resulting SNIR, something not present in the original Tx-ZF. However, the purpose of creating the Generalized Tx-ZF was mainly to get acquainted with the typical constrained optimization problems linked to the design of MIMO linear precoders and filters. Indeed, block diagonalization methods can be shown to provide a much better performance when it comes to zero-forcing schemes, as they make use of many extra degrees of freedom. Generalized Tx-WF will indeed perform better than regular Tx-ZF, because it is a generalization, so that their results would be identical in the worst case. However, it will still be vastly outperformed by Spencer's ZF. As a consequence, we believe that it is not too interesting to discuss empirical results obtained when employing Generalized Tx-ZF.

On the other hand, our main contributions on the field of MIMO linear precoding were directed towards the design of MMSE precoders tailored for the specific needs of cellular coodinated systems. In this way, we tried to enhance existing techniques in two different directions. Firstly, we tried to fill the gap caused by the lack of MMSE precoders with PBPC by attempting to solve MMSE optimization problems under those novel constraints. Doing so represents an essential step essential if we want MMSE precoders which are applicable to multi-user distributed MIMO scenarios such as any future mobile communications system. Alternatively, we introduced the interference-awareness concept creating MMSE precoders which, apart from optimizing the intra-cluster MSE, include in their formulation a minimization of the inter-cluster interference generated by each particular cluster in the system.

Sadly, as we discussed in chapter 3, our proposed MMSE precoders with PBPC are half-finished since we reached analytical solutions characterized by implicit matrix equations whose solutions have not been found yet. To make things even worse, in order to evaluate those implicit equations, we needed to compute the inverse of several pretty big matrices whose elements take very small values. Then, tackling the problem with numerical solvers was out of question as they were unable to converge to the correct solution. Because of that, there is not much we can do to provide empirical results which would allow us discussing the performance of our proposed MMSE precoders for distributed multi-user MIMO scenarios. Thankfully, we can still put to practice our interference-aware MMSE precoder in a scenario without base-station coordination for signal processing, that is, we can try the interference aware Tx-WF of section 3.4.6. Even though that precoder is not applicable in mobile communications, as it does not consider PBPC, we believe that the achieved results will be clear enough to motivate researchers to look for a solution, exact or approximated, to the implicit equation governing the PBPC version of such precoder derived in section 3.4.7.

In order to carry out those simulations, we will employ a context quite similar to that of chapter 6. The simulation scenario will essentially be a shrunk version of the one we employed to illustrate our base-station clustering algorithms. We will consider only 1 tier, hence the base-station deployment only contains 7 cells, as depicted in figure C.1. Moreover, as we cannot resort to any form of PBPC, we have no other choice but to consider singleton clusters, that is, we assume that there is no coordination between cells in the system for joint MIMO transmission.

In this way, we will simulate $N_{\text{rep}} = 50000$ different user deployments within the 7-cell network. For every set of user locations, corresponding to a different channel matrix $\mathbf{H}$, each cell will independently obtain its own precoder $\mathbf{W_{tx}}$ and receive filter $\mathbf{W_{rx}}$ according to both the standard Wiener precoder described in section 3.4.4, Tx-WF; and the scheme we developed, denoted as interference-aware Tx-WF of section 3.4.6. For the latter, even though each cell computes $\mathbf{W_{tx}}$ and $\mathbf{W_{rx}}$ on its own, we will assume that the complete channel matrix is known. In this way, each cell has access to the channel seen towards users in the rest of cells as required by the formulation of the interference-aware Tx-WF.

All other parameters, like those regarding the cell radius or the signal propagation are identical to those considered in chapter 6. That is, we still use the 3GPP path-loss model for LTE in urban environments with additional Rayleigh fading. The signal-to-noise ratio at the cell border is $\rho = 15$ dB and the cell radius $R_{\text{cell}} = 250m$.

In this simulation, we do change the MIMO configuration by including a wider variety of settings. We include the results attained when using not only $t = 2, r = 2$ as in chapter 6, but also $t = 4, r = 2$, $t = 4, r = 4$, $t = 8, r = 2$, $t = 8, r = 4$ and $t = 8, r = 8$.

In table C.1 we show the results of our simulation.

A graphical representation of the previous results is shown in figure C.2. There, we plot the cumulative density functions of the user rate R and per-user MSE resulting for the Wiener precoder with and without interference-awareness for each of the six particular MIMO configu-

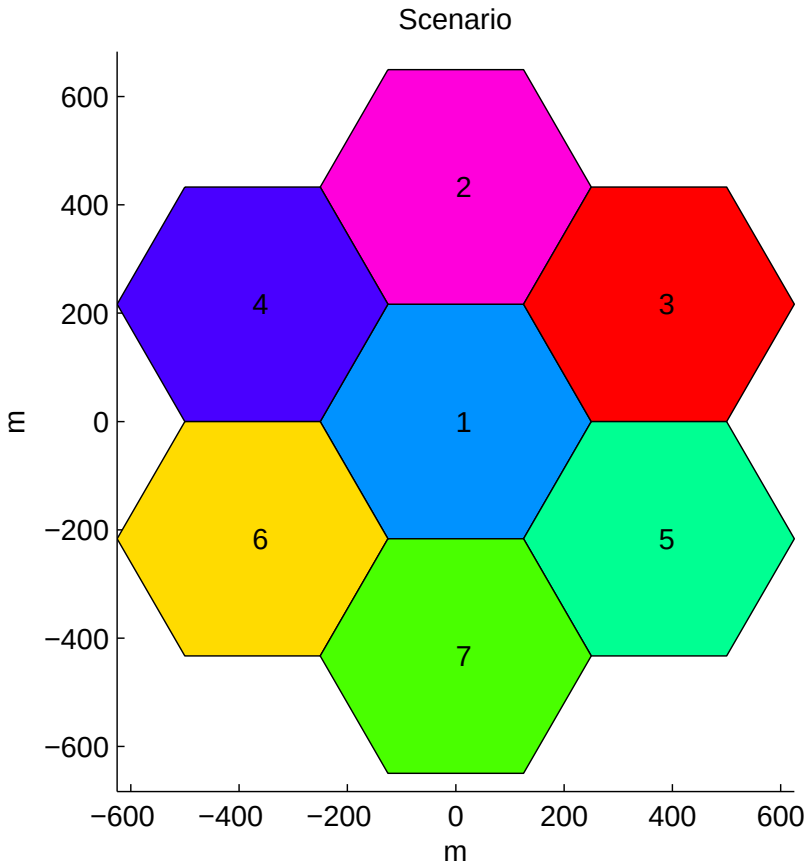


Figure C.1: BTS deployment used to illustrate the Interference-aware Tx-WF precoder

rations we have simulated.

The results are simply devastating for the standard Wiener precoder. According to table C.1, the introduction of interference-awareness can provide us with increases of up to 50% in the median user rate and 35% in the average user rate. Not only that, the MSE per user can be reduced up to 75%, figures which without any doubt surpass even our most optimistic expectations.

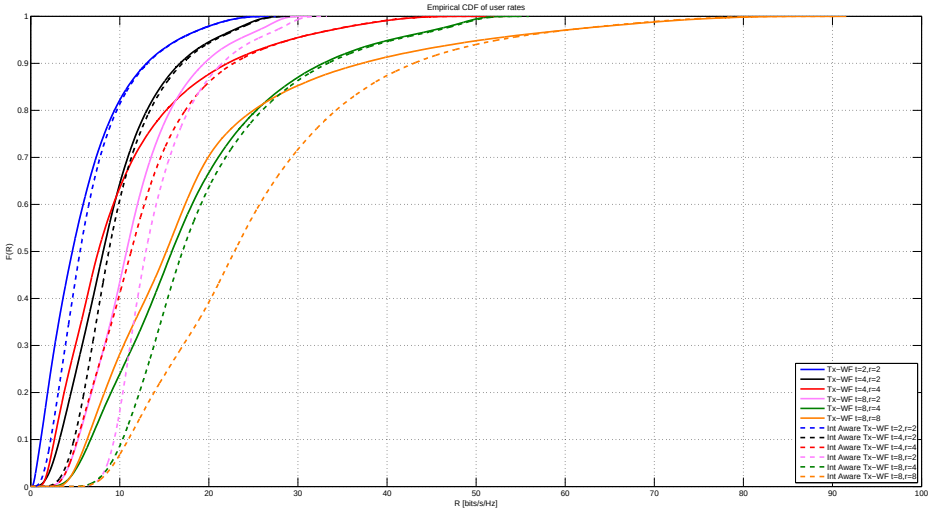
In figure C.2, we have a clearer picture of how big the existing gap between the interference-aware Tx-WF and the regular Tx-WF is. We can also appreciate that the improvement margin increases with the number of antennas and, moreover, is specially big whenever there are as many transmit antennas as receive antennas in each cell.

To sum up, even if this simulation is certainly quite limited, we believe that the interference-awareness concept we developed during this project could not be more promising. We strongly believe that, according to this results, the interference-aware Tx-WF precoder has enough potential to outperform even current block diagonalization methods. Therefore, this acts as a strong source of motivation for trying to solve the PBPC problem to be ready for testing our brand new PBPC interference-aware Wiener precoder in a realistic, cellular coordinated MIMO

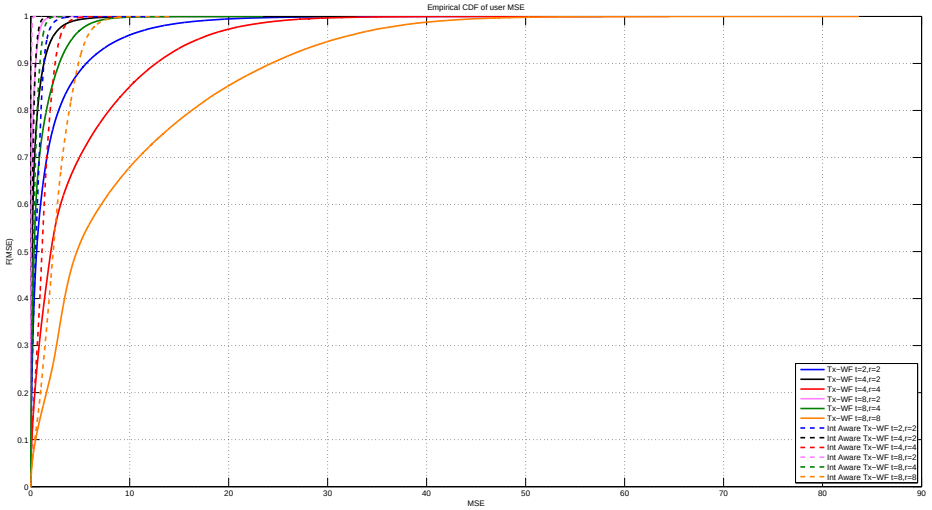
Configuration		Tx-WF	Interference-Aware Tx-WF
$\boldsymbol{t=2, r=2}$	$\overline{R}$	6,016	6,744
	$\widetilde{R}$	4,677	5,544
	$\overline{\mathsf{MSE}}$	1,960	0,618
	$\widetilde{\mathsf{MSE}}$	0,599	0,495
$\boldsymbol{t=4, r=2}$	$\overline{R}$	9,169	9,977
	$\widetilde{R}$	8,141	8,765
	$\overline{\mathsf{MSE}}$	0,481	0,196
	$\widetilde{\mathsf{MSE}}$	0,147	0,116
$\boldsymbol{t=4, r=4}$	$\overline{R}$	10,275	12,904
	$\widetilde{R}$	7,586	11,273
	$\overline{\mathsf{MSE}}$	4,464	1,245
	$\widetilde{\mathsf{MSE}}$	2,040	1,140
$\boldsymbol{t=8, r=2}$	$\overline{R}$	11,659	14,296
	$\widetilde{R}$	10,757	12,933
	$\overline{\mathsf{MSE}}$	0,169	0,036
	$\widetilde{\mathsf{MSE}}$	0,056	0,024
$\boldsymbol{t=8, r=4}$	$\overline{R}$	18,013	19,688
	$\widetilde{R}$	15,851	17,042
	$\overline{\mathsf{MSE}}$	0,965	0,386
	$\widetilde{\mathsf{MSE}}$	0,342	0,270
$\boldsymbol{t=8, r=8}$	$\overline{R}$	18,789	25,394
	$\widetilde{R}$	15,220	22,877
	$\overline{\mathsf{MSE}}$	9,051	2,499
	$\widetilde{\mathsf{MSE}}$	4,665	2,277

Table C.1: Comparision of average and median values of the user rate R and per-user MSE *attained by Tx-WF and interference-aware Tx-WF for each particular MIMO configuration.*

system.



(a) User rates



(b) MSE

Figure C.2: CDF of the user rate R and per-user MSE *attained by Tx-WF and interference-aware Tx-WF for each particular MIMO configuration.*

Appendix D

Budget

In this appendix we present the budget which justifies the cost incurred by this project. Those are mainly subdivided in personnel cost and material costs.

In tables D.1 and D.2 we show an estimation of the time devoted to each of the four main phases of the project by its contributors. The first table accounts for the work carried out by the author of the project, which is still an undergraduate student. On the other hand, the second table contains an approximation of the number of hours devoted by the two supervisors of this project, telecommunication engineers with Ph. D.

Table D.1: Time dedicated to each phase of the project by its author

Phase 1	*Previous study*	150 *hours*
Phase 2	*Theoretical derivations*	400 *hours*
Phase 3	*Implementation in MATLAB*	500 *hours*
Phase 4	*Documentation*	500 *hours*

Table D.2: Time dedicated to each phase of the project by the supervisors

Phase 1	*Previous study*	25 *hours*
Phase 2	*Theoretical derivations*	100 *hours*
Phase 3	*Implementation in MATLAB*	5 *hours*
Phase 4	*Documentation*	100 *hours*

The total number of hours dedicated by its author have been approximately 1550. Considering a typical undergraduate salary of 6 € per hour, the author's compensation would amount to 9.300 €. Besides, the supervisors have employed around 230 hours in this project. Assuming a salary in the order of 25 € per hour, their compensation amounts to 5750 €. Hence, the total

Table D.3: Material costs

Laptop Asus N61JQ	1.100 €
MATLAB license for faculty, staff, or researcher	500 €
Others	50 €

Table D.4: Budget

Concept	**Amount**
Personnel costs	15.050 €
Material costs	1.650 €
Tax base	16.700 €
VAT (21%)	3.507 €
TOTAL	20.207 €

personnel costs are 15050 €.

In table D.3 we include the main material costs of this project. As most of our work has been theoretical, it merely amounts to the computational system required to carry out our simulations: a personal computer with a MATLAB license. As we can see, the total material cost has been 1.650 €.

Putting all together, the complete budget is shown in table D.4. We can observe that most of our budget is derived from personnel costs.

The total cost of the project amounts to **TWENTY THOUSAND, TWO HUNDRED AND SEVEN EUROS (20.207 €)**.

Bibliography

[1] A. Brazma and J. Vilo, "Gene expression data analysis," *FEBS Lett*, vol. 480, pp. 17–24, 2000.

[2] U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, pp. 395–416, Dec 2007.

[3] Z. Zhang and M. I. Jordan, "Multiway Spectral Clustering: A Margin-Based Perspective," *Statistical Science*, vol. 23, pp. 383–403, 2008.

[4] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888–905, Aug 2000.

[5] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances In Neural Information Processing Systems*, pp. 849–856, MIT Press, 2001.

[6] S. Y. Jianbo, S. X. Yu, and J. Shi, "Multiclass spectral clustering," in *In International Conference on Computer Vision*, pp. 313–319, 2003.

[7] C. E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. J.*, 1948.

[8] B. Holter, "On the capacity of the mimo channel - a tutorial introduction," in *IEEE Vehicular Technologies Conference (VTC'98)*, vol. 1, pp. 615 – 619, May 1998.

[9] M. Joham, W. Utschick, and J. Nossek, "Linear transmit processing in MIMO communications systems," *IEEE Transactions on Signal Processing*, vol. 53, pp. 2700 – 2712, Aug. 2005.

[10] D. P. Palomar, "Joint tx-rx beamforming design for multicarrier mimo channels: a unified framework for convex optimization," *IEEE Transactions on Signal Processing*, vol. 51, pp. 2381–2401, Sep 2003.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[12] D. P. Bertsekas, *Convex Analysis and Optimization*. Athena Scientific, 2003.

[13] D. P. Palomar and Y. Jiang, "Mimo transceiver design via majorization theory," *Found. Trends Commun. Inf. Theory*, vol. 3, pp. 331–551, nov 2006.

[14] A. Hjorungnes and D. Gesbert, "Complex-valued matrix differentiation: Techniques and key results," *IEEE Transactions on Signal Processing*, vol. 55, pp. 2740–2746, June 2007.

[15] Q. Spencer and A. S. M. Haardt, "Zero-forcing methods for downlink spatial multiplexing in multiuser MIMO channels," *IEEE Transactions on Signal Processing*, vol. 52, pp. 461 – 4716, Feb. 2004.

[16] G. Foschini, K. Karakayali, and R. Valenzuela, "Coordinating multiple antenna cellular networks to achieve enormous spectral efficiency," *IEEE Proceedings-Communications*, vol. 153, pp. 548–555, Aug. 2006.

[17] D. Lay, *Linear Algebra and Its Applications*. Pearson Education, 2002.

[18] A. García, M. P. Sánchez, and R. Corvaja, "Constrained power allocation schemes for coordinated base station transmission using block diagonalization," *EURASIP Journal on Wireless Communication and Networking*, Oct. 2011.

[19] N. Wiener, *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. Technology press books in science and engineering, Technology Press of the Massachusetts Institute of Technology, 1949.

[20] K. B. Petersen and M. S. Pedersen, "The Matrix Cookbook," Oct. 2008.

[21] W. M. Rand, "Objective Criteria for the Evaluation of Clustering Methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.

[22] D. Jians, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp. 1370 – 1386, Nov 2004.

[23] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," *In Procedings of the Fifth Berkeley Symposium on Math, Statistics, and Probability*, vol. 1, pp. 281–297, 1967.

[24] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, Mar 1982.

[25] C. Ding and X. He, "Cluster merging and splitting in hierarchical clustering algorithms," in *Proc. IEEE Int'l Conf. Data Mining*, pp. 139–146, 2002.

[26] B. Mohar, "The laplacian spectrum of graphs," in *Graph Theory, Combinatorics, and Applications*, pp. 871–898, Wiley, 1991.

[27] M. Stoer and F. Wagner, "A simple min-cut algorithm," *Journal of the ACM*, vol. 44, pp. 585–591, Jul 1997.

[28] D. Wagner and F. Wagner, "Between min cut and graph bisection," in *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*, MFCS '93, pp. 744–750, 1993.

[29] J. Gower and G. B. Dijksterhuis, *Procrustes Problems*. Oxford University Press, 2004.

[30] F. R. Bach and M. I. Jordan, "Learning spectral clustering, with application to speech separation," *Journal Of Machine Learning Research*, vol. 7, p. 2006, 2006.

[31] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, vol. 21, pp. 32–40, Jan 1975.

[32] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 790–799, Aug 1995.

[33] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, May 2002.

[34] E. Parzen, "On estimation of a probability density function and mode," *Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[35] M. Lázaro-Gredilla, V. Gómez-Verdejo, and E. Parrado-Hernández, "Low-cost model selection for svms using local features," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 6, pp. 1203 – 1211, 2012.

[36] V. A. Epanechnikov, "Non parametric estimation of a multivariate probability density," *Theory of Probability and Its Applications*, vol. 14, pp. 153–158, 1969.

[37] X. Liand, Z. Hu, and F. Wu, "A note on the convergence of the mean shift," *Pattern Recognition*, vol. 40, pp. 1756–1762, Jun 2007.

[38] D. Comaniciu and P. Meer, "Distribution free decomposition of multivariate data," *Pattern Analysis and Applications*, vol. 2, pp. 22–30, 1998.

[39] U. Ozertem, D. Erdogmus, and R. Jenssen, "Mean shift spectral clustering," *Pattern Recognition*, vol. 41, pp. 1924–1938, jun 2008.

[40] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[41] D. Whitley and N. wook Yoo, "Modeling simple genetic algorithms for permutation problems," in *in Foundations of Genetic Algorithms*, pp. 163–184, Morgan Kaufmann, 1995.

[42] E. Hruschka, R. Campello, A. Freitas, and A. de Carvalho, "A survey of evolutionary algorithms for clustering," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 39, pp. 133 –155, Mar 2009.

[43] E. Falkenauer, *Genetic Algorithms and Grouping Problems*. John Wiley & Sons, Inc., 1998.

[44] H. Wang, J. Chen, and K. Guo, "A genetic spectral clustering algorithm," *Journal of Computational Information Systems*, vol. 7, no. 9, pp. 3245–3252, 2011.

[45] K. Krishna and M. Narasimha Murty, "Genetic k-means algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, pp. 433–439, Jun 1999.

[46] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S. J. Brown, "Fgka: a fast genetic k-means clustering algorithm," in *Proceedings of the 2004 ACM symposium on Applied computing*, SAC '04, pp. 622–623, ACM, 2004.

[47] J. Xiao, Y. Yan, J. Zhang, and Y. Tang, "A quantum-inspired genetic algorithm for k-means clustering," *Expert Syst. Appl.*, vol. 37, pp. 4966–4973, Jul 2010.

[48] V. Alves, R. Campello, and E. Hruschka, "Towards a fast evolutionary algorithm for clustering," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pp. 1776 –1783, 2006.

[49] R. Tarjan, "Depth-First Search and Linear Graph Algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.

[50] M. Meila and J. Shi, "A random walks view of spectral segmentation," *In Proceedings of 8th International Workshop on Artificial Intelligence and Statistics*, 2001.

[51] N. Cristianini and J. Shawe-Taylor, *An introduction to support Vector Machines: and other kernel-based learning methods*. New York, NY, USA: Cambridge University Press, 2000.

[52] U. N. L. o. M. Lister Hill National Center for Biomedical Communications, "Handbook: Help Me Understand Genetics." http://ghr.nlm.nih.gov/handbook/, 2012. [Online; accessed 14-August-2012].

[53] S. E. DeWeerdt. http://www.genomenewsnetwork.org/, 2003. [Online; accessed 14-August-2012].

[54] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Molecular biology of the cell*. Garland Science Taylor & Francis Group, 5 ed., 2007.